LAB conventions

Before starting, it is important to review some lab conventions that will ease your work…

- **Lab steps are in black and 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 ?

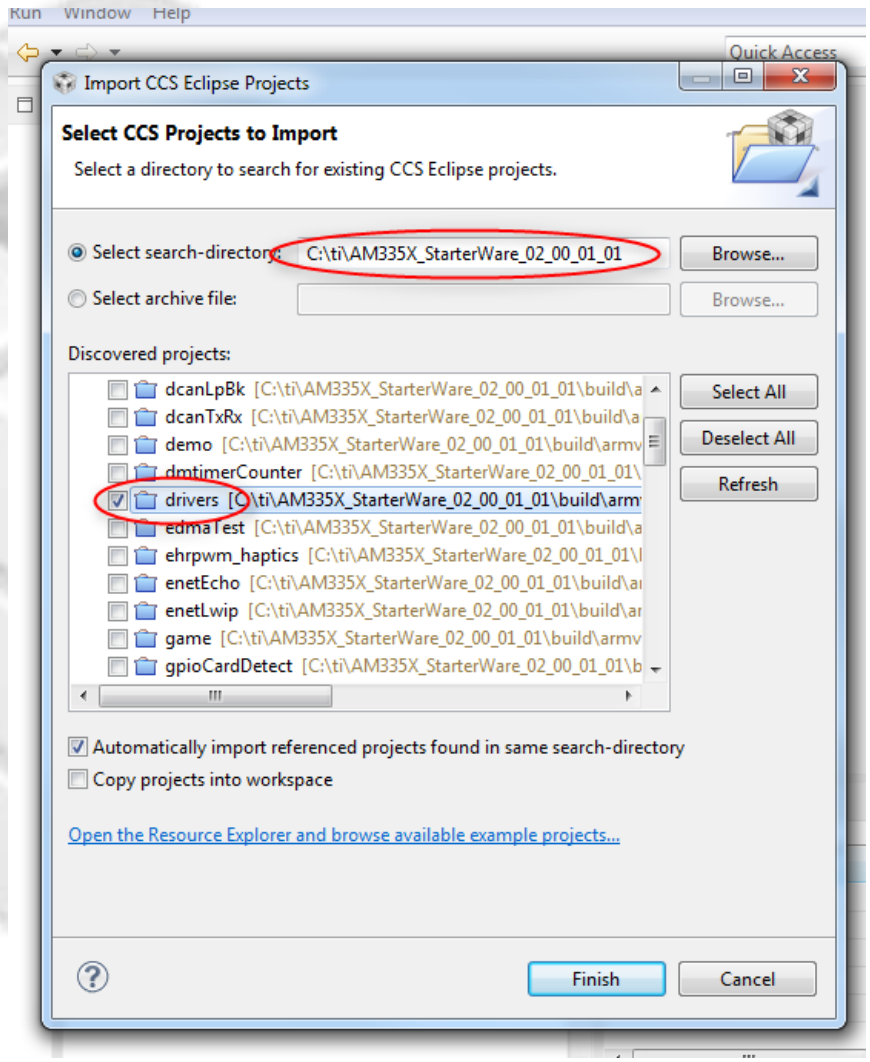
GPIO LED Blink Example: Exercise Summary

- **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

Import the support library projects

1. Import the **drivers, system and utils** projects into the CCS Workspace by going to menu *Project → Import CCS Projects*
2. In the box *Select search-directory*, type or browse to the Starterware installation directory:

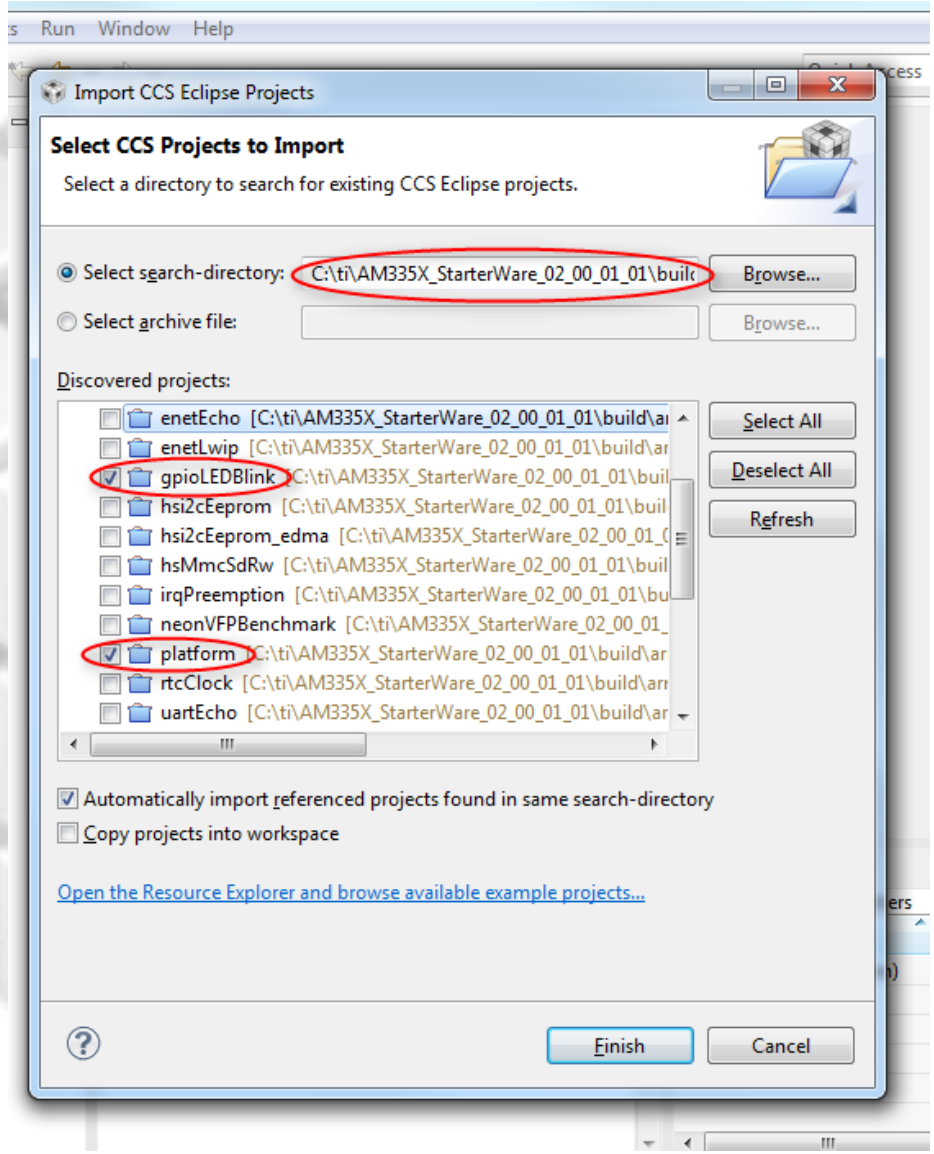
   C:\TI\AM335X_StarterWare_02_00_01_01
3. Wait until the tool finishes discovering the available projects.
4. Select the projects **drivers, system and utils**
5. Click *Finish*



Import the 'platform' and 'gpioLEDBlink' projects

1. Import the **platform and gpioLEDBlink** projects into the CCS Workspace by going to menu *Project* → *Import CCS Projects*
2. In the box *Select search-directory*, type or browse to the Beaglebone builds directory:

   C:\TI\AM335X_StarterWare_02_00_01_01\build\armv7a\cgt_ccs\am335x\beaglebone
3. Wait until the tool finishes discovering the available projects.
4. Select the projects **platform** and **gpioLEDBlink**
5. Click *Finish*



Build the support library projects

1. In the *Project Explorer* view, select the support library projects by hitting Ctrl key and clicking each with the left mouse button.

2. Click on the Build icon in the toolbar. The project will start the building process.

   When using a newly installed CCS, the tool will take extra time to build the Runtime Support Library (RTS) at this time. This is normal and will only happen once.

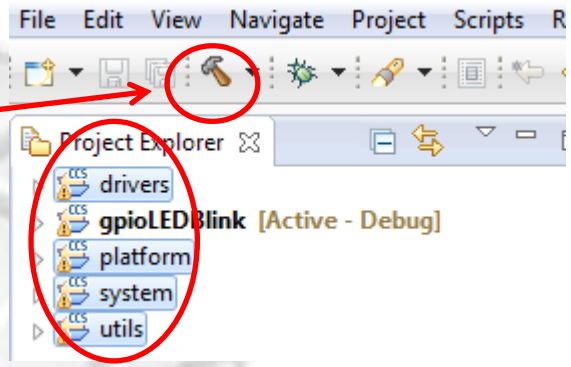
3. The *Console* view will appear at the bottom with build messages (information, warnings, errors) as the project builds

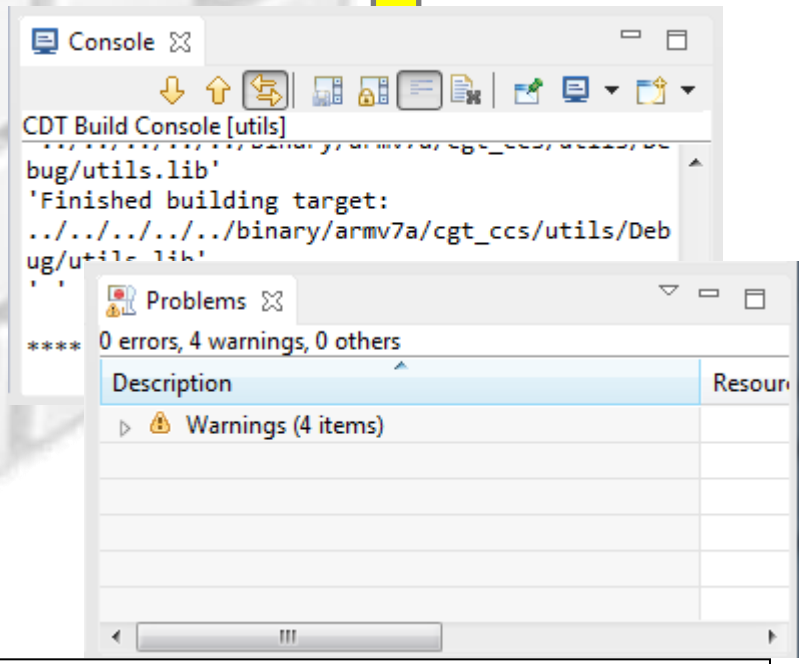
4. The *Problems* view will also appear at the bottom to highlight any possible build errors.

   When building the RTS, some warning messages will appear in the problems view and can be ignored.

   Warnings also will appear due to the difference in compiler tools used originally by Starterware

5. If the build is successful, the *Problems* view will contain no errors (warnings can still be seen)





Build ‘gpioBlinkLED’ Project

1. In the *Project Explorer* view, select the **gpioLEDBlink** project (it should appear **[Active – Debug]**) to make it active

2. Click on the Build icon in the toolbar. The project will start the building process.

   When using a newly installed CCS, the tool will take extra time to build the Runtime Support Library (RTS) at this time. This is normal and will only happen once.

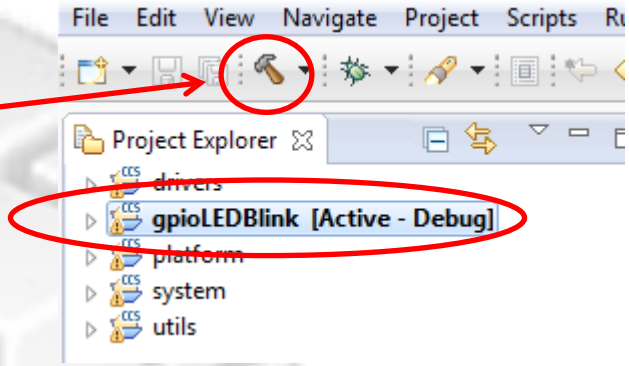
3. The *Console* view will appear at the bottom with build messages (information, warnings, errors) as the project builds

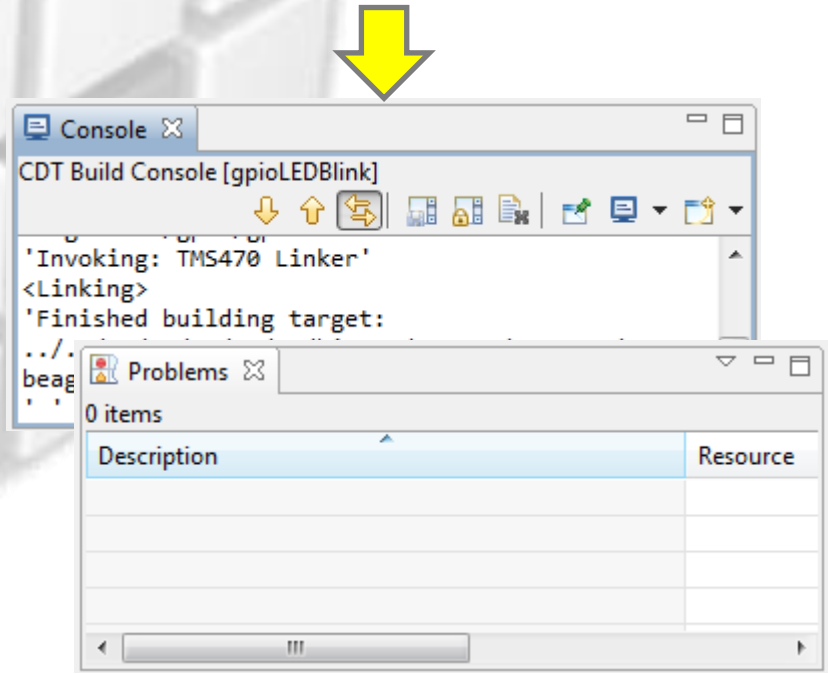
4. The *Problems* view will also appear at the bottom to highlight any possible build errors.

   When building the RTS, some warning messages will appear in the problems view and can be ignored.

   Warnings also will appear due to the difference in compiler tools used originally by Starterware

5. If the build is successful, the *Problems* view will contain no errors (warnings can still be seen)





Create a Target Configuration File

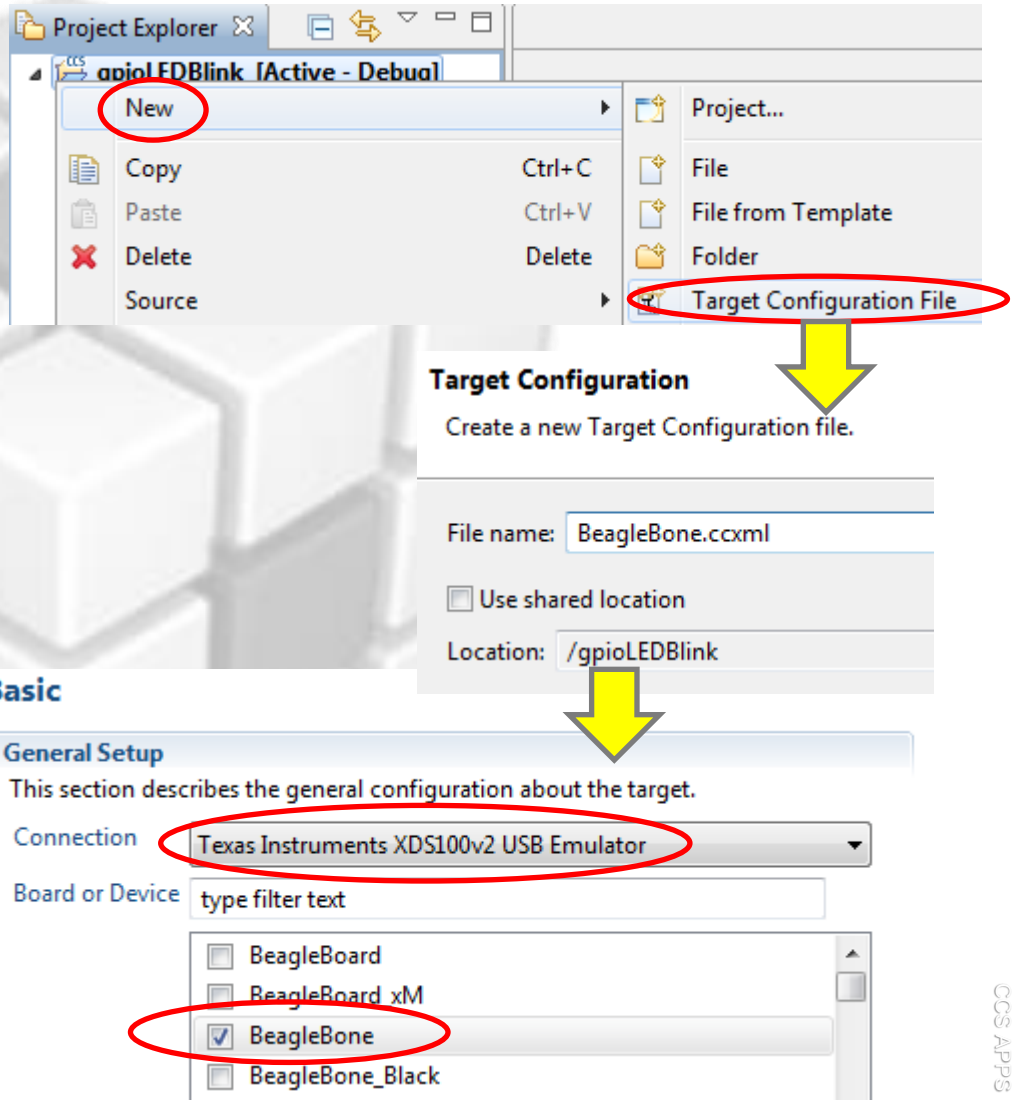
In order to connect to the board, a Target Configuration File must be created. This file contains the information about the JTAG emulator and the device to be connected.

1. In the *Project Explorer*, right-click on the project and select *New* → *Target Configuration File*. A dialog box will be shown - give *BeagleBone* as a name and click *Finish*

The target configuration editor will be shown

2. For the *Connection* choose *Texas Instruments XDS100v2 USB Emulator*

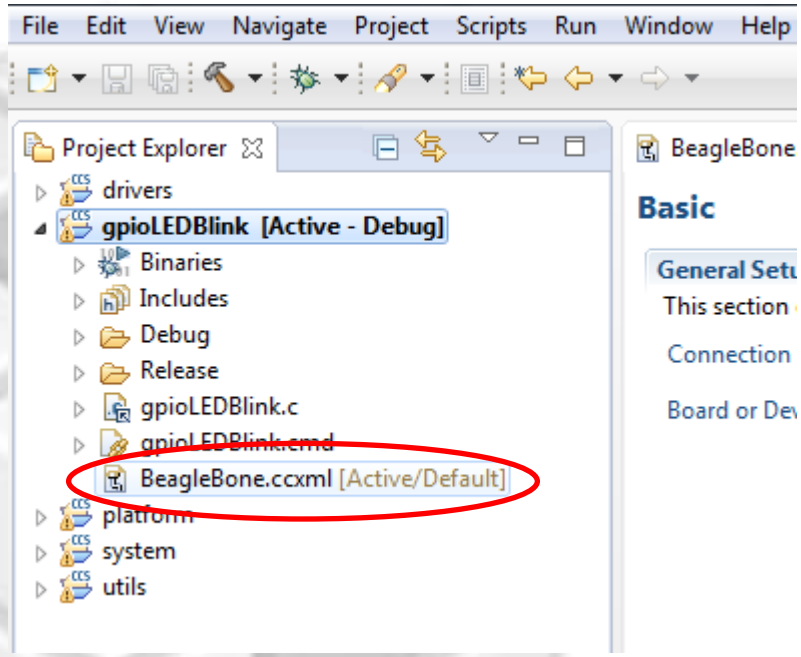
3. For the *Board or Device* choose *BeagleBone*



4. Click *Save*

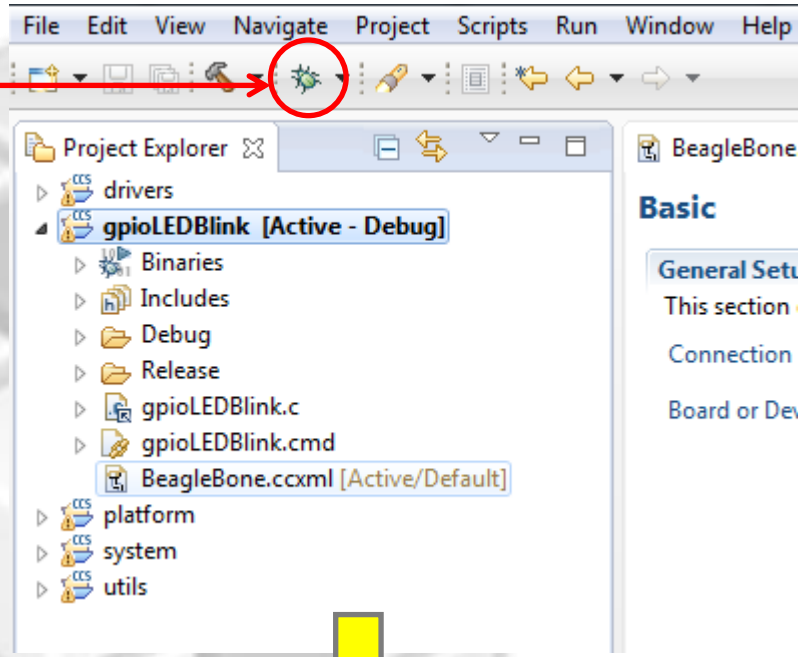
Debugger is ready for launch

- The Target Configuration File will be added to the project
- The [Active/Default] indicates this is the target configuration that will be used to debug this project (Active) and is also the Default for all other projects of the workspace (unless they have active configurations of their own).



Debug ‘gpioBlinkLED’ Project

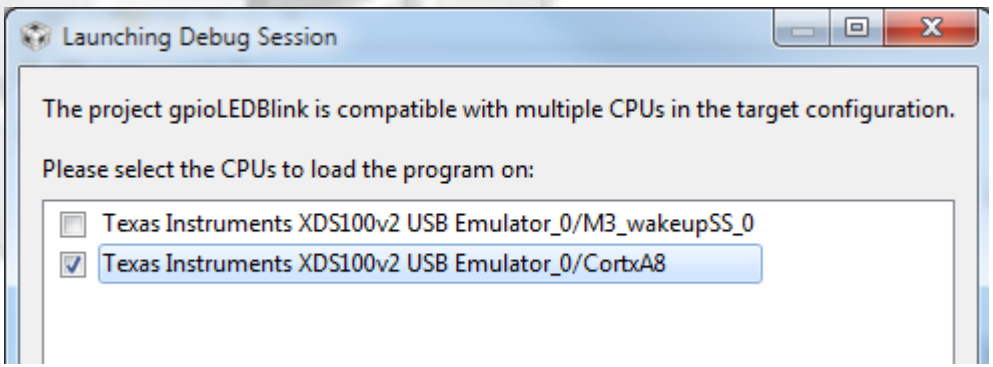
1. Click on the “green bug” button – make sure the project is selected!
2. When the Debug Session is launching, CCS asks which cores to load. De-select **M3_wakeupSS_0**



This happens because BeagleBone has two cores compatible with the project (ARM cores)

When you hit the green bug button, several actions are done automatically

- Prompt to save source files
- Build the project (incrementally)
- Start the debugger (CCS will switch to the *CCS Debug* perspective)
- Connect CCS to the target
- Load the program on the target
- Run to **main()**



Debug ‘gpioBlinkLED’ Project

CCS Debug - gpioLEDBlink/gpioLEDBlink.c - Code Composer Studio

File Edit View Project Tools Scripts Run Window Help

Quick Access | CCS Edit | CCS Debug

Debug

gpioLEDBlink [Code Composer Studio - Device Debugging]
- Texas Instruments XDS100v2 USB Emulator_0/M3_wakeupSS_0 (Disconnected : Unknown)
- Texas Instruments XDS100v2 USB Emulator_0/CortxA8 (Suspended - SW Breakpoint)
  - main() at gpioLEDBlink.c:83 0x800015AC
  - start_boot() at startup.c:143 0x80001658
  - $C:/ti/AM335X_StarterWare_02_00_01_01/system_config/armv7a/cgt/init.asm:97:190$() a

Variables | Expressions | Registers

Name	Type	Value	Location

Switched to *CCS Debug* perspective

BeagleBone.ccxml | gpioLEDBlink.c

```
76 /*****************************************************************************
77 **                INTERNAL FUNCTION DEFINITIONS
78 *****************************************************************************/
79 /*
80 ** The main function. Application starts here.
81 */
82 int main()
83 {
84     /* Enabling functional clocks for GPIO1 instance. */
85     GPIO1ModuleClkConfig();
86
87     /* Selecting GPIO1[23] pin
88     GPIO1Pin23PinMuxSetup();
89
90     /* Enabling the GPIO module. */
91     GPIOModuleEnable(GPIO_INSTANCE_ADDRESS);
92
93     /* Resetting the GPIO module. */
```

Program counter at main()

Console

gpioLEDBlink

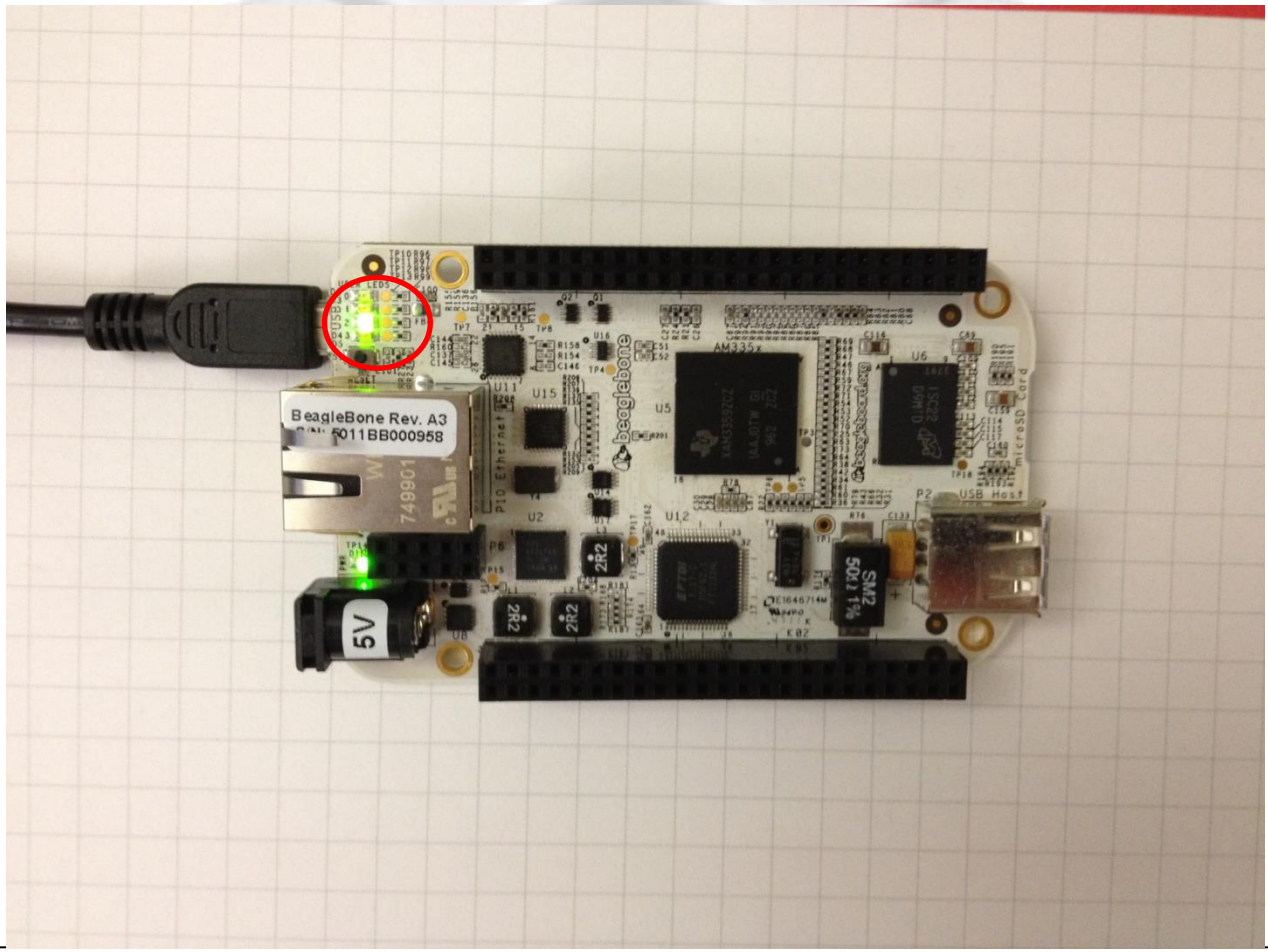
```
CortxA8: Output: EMIF Timing register configuration is in progress .......
CortxA8: Output: EMIF Timing register configuration is done .......
CortxA8: Output: PHY is READY!!
CortxA8: Output: DDR PHY Configuration done
CortxA8: Output: ****  AM335x Beaglebone Initialization is Done *****************
```

Writable | Smart Insert | 83 : 2 | Full License | ARM | LE | SYS | MMU Off

Blink USR2 LED (D4)

1. **Press the *Run* button to run the program**

USR2 LED on the BeagleBone should now be blinking

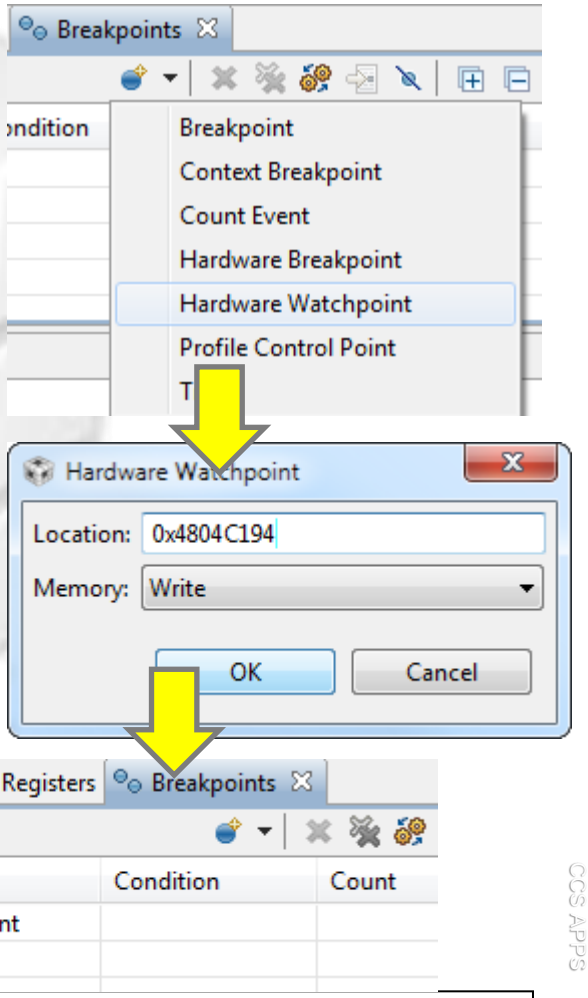


Debugging: Using Watchpoints

A Watchpoint is a type of breakpoint that monitors activity on a memory address

In this step we will set a watchpoint to halt the CPU anytime the LED will toggle

1. **Press the *halt/suspend* button to halt the running program**
   - The code should stop either in the **Delay()** function or somewhere inside the **while(1)** loop
2. **Open the *Breakpoints* view**
   - *View → Breakpoints*
3. **Create a new *Hardware Watchpoint***
   - *Location*: set it to 0x4804C194 (**GPIO_SETDATAOUT** register)
   - *Memory*: *Write*



Debugging: Using Watchpoints

1. Run the target again. Execution will automatically halt when USR2 LED is toggled (the GPIO_SETDATAOUTPUT register is written to)

The program stops at GPIOPinWrite(), which is part of the drivers library

CCS Debug - drivers/gpio_v2.c - Code Composer Studio

File Edit View Project Tools Scripts Run Window Help

Quick Access | CCS Edit | CCS Debug

Debug

gpioLEDBlink [Code Composer Studio - Device Debugging]
- Texas Instruments XDS100v2 USB Emulator_0/M3_wakeupSS_0 (Disconnected : Unknown)
- Texas Instruments XDS100v2 USB Emulator_0/CortxA8 (Suspended - HW Breakpoint)
  - GPIOPinWrite(unsigned int, unsigned int, unsigned int)() at gpio_v2.c:192 0x8000021C
  - main() at gpioLEDBlink.c:107 0x800015E8
  - start_boot() at startup.c:143 0x80001658

Variables | Expressions | Registers | Breakpoints

Identity	Name	Condition	Count	Action
0x4804C194 (G	Watchpoint			

BeagleBone.ccxml | gpioLEDBlink.c | gpio_v2.c

```
186 void GPIOPinWrite(unsigned int baseAdd,
187                  unsigned int pinNumber,
188                  unsigned int pinValue)
189 {
190     if(GPIO_PIN_HIGH == pinValue)
191     {
192         HWREG(baseAdd + GPIO_SETDATAOUT) = (1 << pinNumber);
193     }
194     else
195     {
196         HWREG(baseAdd + GPIO_CLEARDATAOUT) = (1 << pinNumber);
197     }
198 }
199
200 /**
201  * \brief   This API determines the logic level(value) on a specified
202  *          GPIO pin.
```

Console

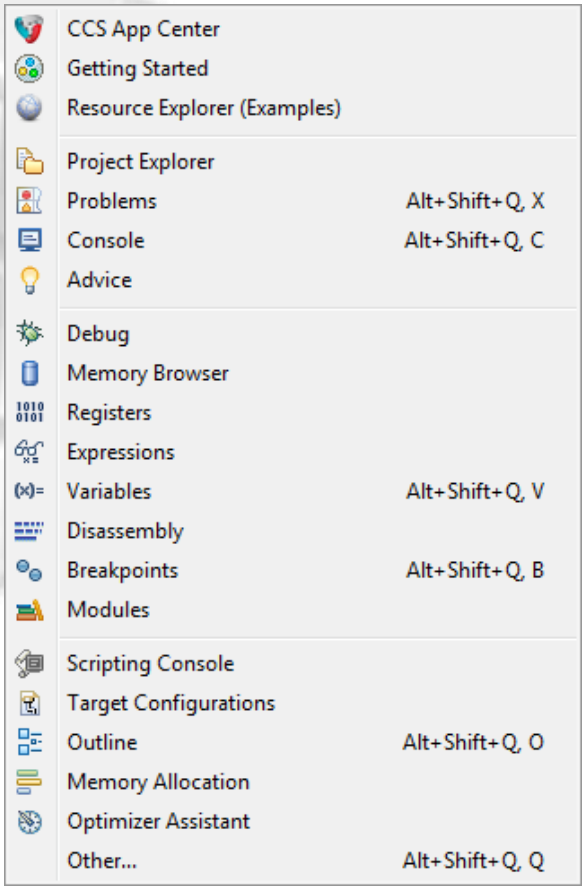
gpioLEDBlink

```
CortxA8: Output: EMIF Timing register configuration is in progress .......
CortxA8: Output: EMIF Timing register configuration is done .......
CortxA8: Output: PHY is READY!!
CortxA8: Output: DDR PHY Configuration done
CortxA8: Output: ****  AM335x Beaglebone Initialization is Done *****************
```

Writable | Smart Insert | 192 : 1 | Full License | ARM | LE | SYS | MMU Off

More Debugging

- **Investigate other debugging views (Open via *View* menu)**
  - Memory Browser
  - Registers
  - Disassembly (see next slide)
- **Set breakpoints**
  - Double click on a source line to set/clear
  - See list of breakpoints with the *Breakpoints* view
- **Use the buttons in the *Debug* view to:**
  - Restart the program
  - Source stepping
  - Assembly stepping

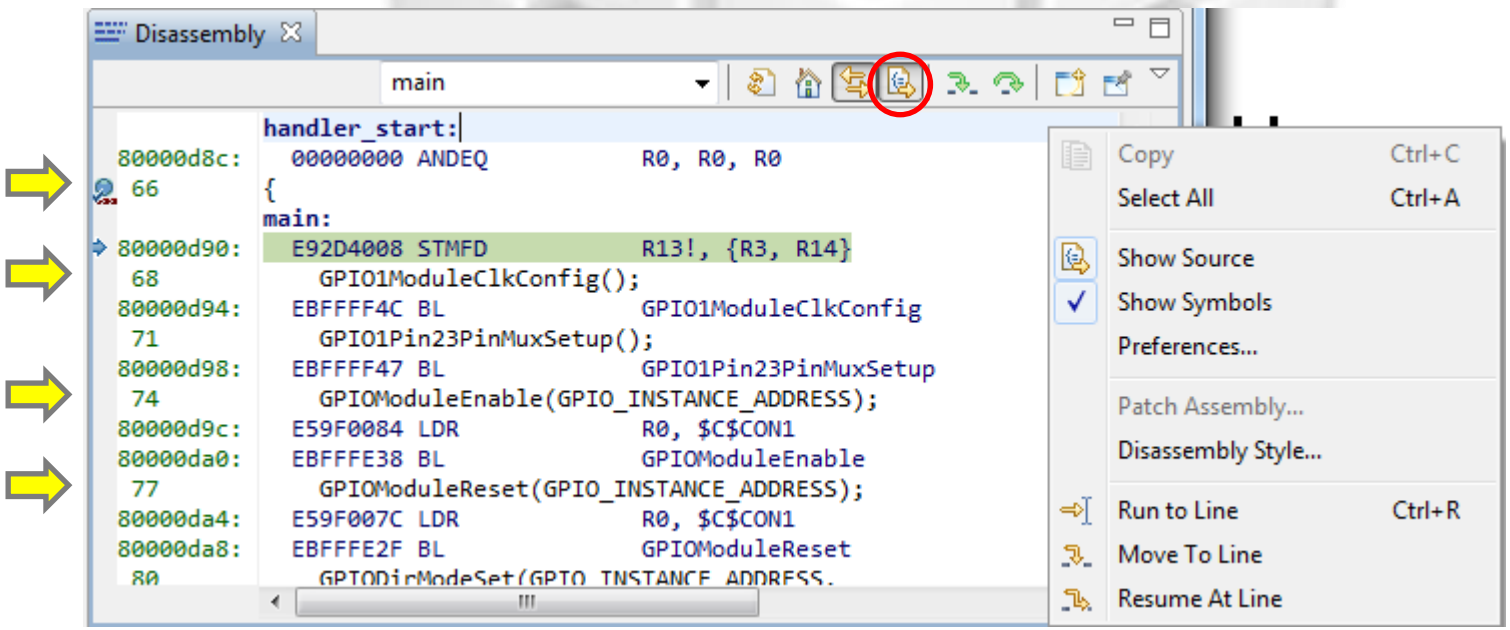


View: Disassembly

1. Open the Disassembly view. Go to the main() symbol in this view by typing “main” in the address field and hit <ENTER>

   Just as an example, the picture below shows the current location of the PC (small blue arrow) and any breakpoints (small blue circles)

2. Toggle the *Show Source* button. Note the toggling of interleaved source with the disassembly



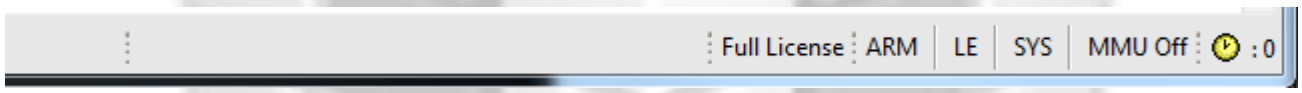
Debugging: Using Profile Clock

The profile clock

- Available on most devices and can be used to count cycles
- On some targets it can be used to count other events like cache hits/misses, bus stalls, etc.

1. Enable the Clock

- Menu *Run* → *Clock* → *Enable*
- The clock will now be displayed on the status bar



Check that the watchpoint that is watching for writes to GPIO_SETDATAOUT is enabled

2. Click the *Run* button

- Clock should now show ~203M cycles

Tip: Double-clicking on the clock icon will reset the count to '0'

Increase USR2 LED Blink Rate

The blink rate of USR2 LED can be increased by changing the value of the delay

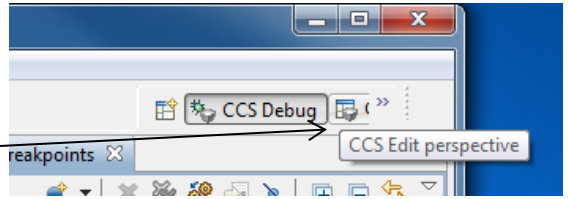
1. **Modify lines 107 and 114 of gpioLEDBlink.c**
   - From: `Delay(0x3FFFF);`
   - To: `Delay(0x1FFFF);`

Beaglebone.ccxml | gpioLEDBlink.c | gpio_v2.c

```
98                  GPIO_INSTANCE_PIN_NUMBER,
99                  GPIO_DIR_OUTPUT);
100     while(1)
101     {
102         /* Driving a logic HIGH on the GPIO pin. */
103         GPIOPinWrite(GPIO_INSTANCE_ADDRESS,
104                      GPIO_INSTANCE_PIN_NUMBER,
105                      GPIO_PIN_HIGH);
106
107         Delay(0x1FFFF);
108
109         /* Driving a logic LOW on the GPIO pin. */
110         GPIOPinWrite(GPIO_INSTANCE_ADDRESS,
111                      GPIO_INSTANCE_PIN_NUMBER,
112                      GPIO_PIN_LOW);
113
114         Delay(0x1FFFF);
115     }
116
117 }
```

2. **Remember to disable the watchpoint set before! Go to the Breakpoints view and uncheck its checkbox**

View: Local History

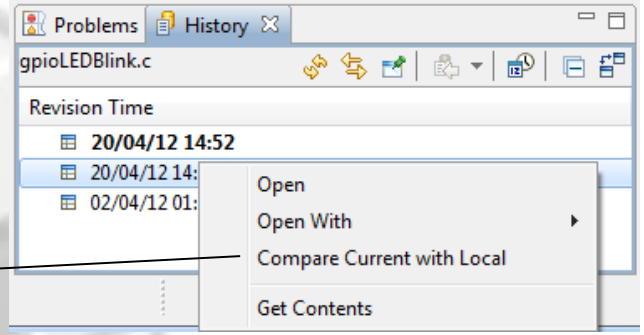


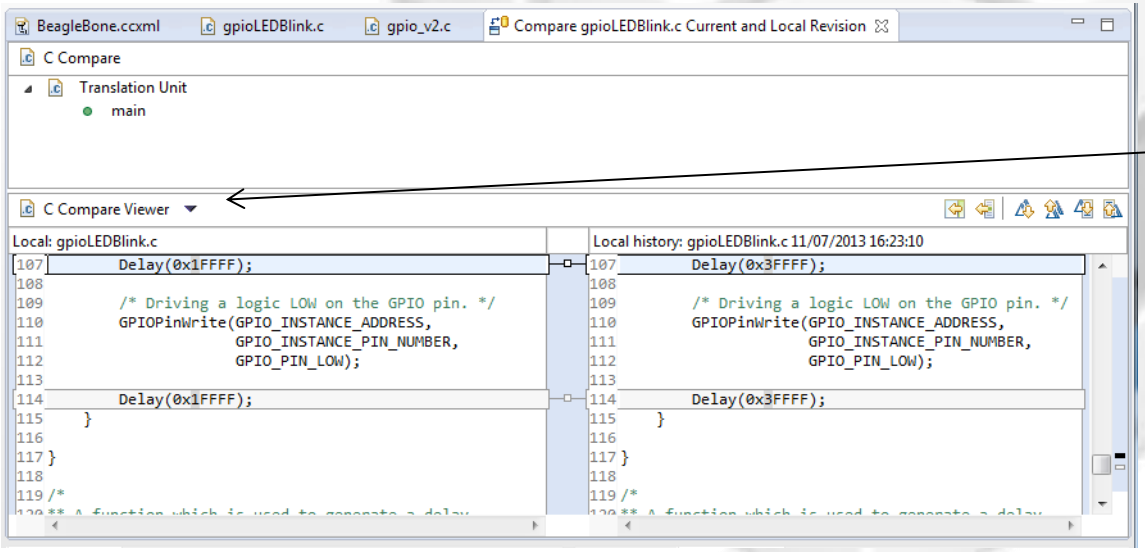
CCS keeps a local history of source changes

1. Switch to the *CCS Edit* perspective
2. Right-click on a file in the editor and select *Team* → *Show Local History*

You can compare your current source file against any previous version or replace it with any previous version

1. Double-click on a revision to open it in the editor
2. Right-click on a revision to compare that revision to the current version





CCS also keeps project history

- If files are deleted from the project, CCS can recover them

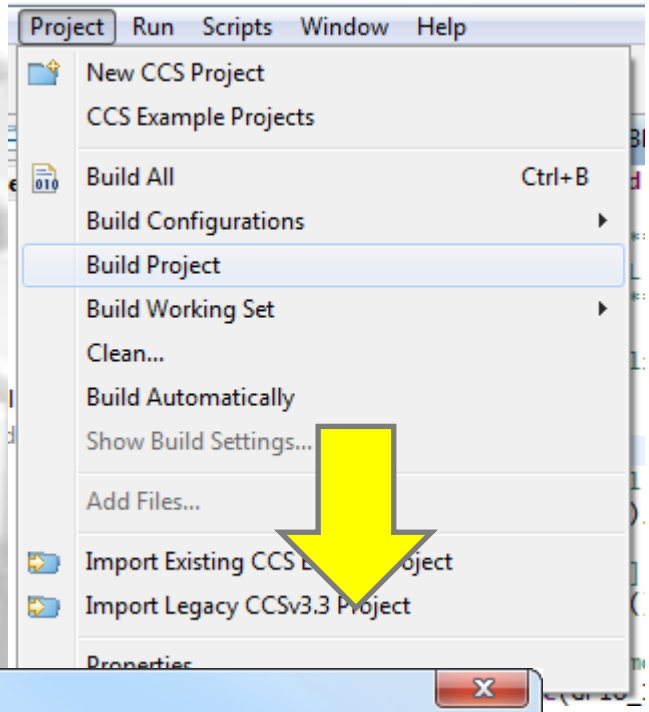
1. Right-click on the project and select *Restore from Local History* in the context menu

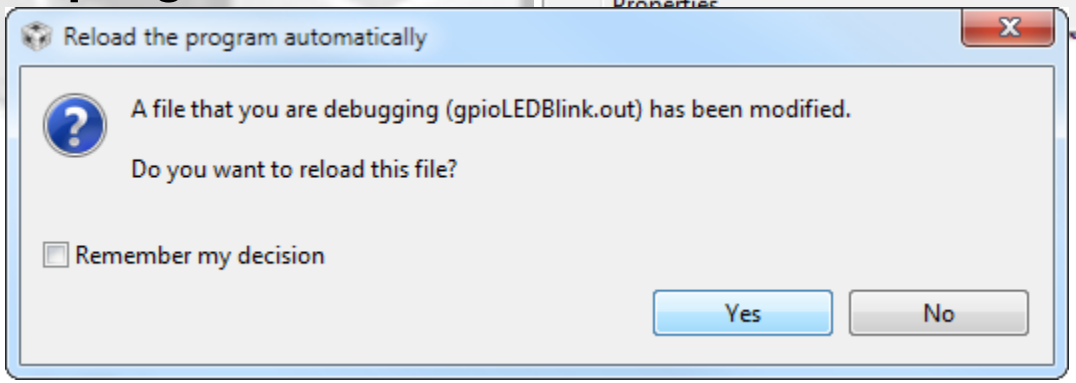
Rebuild Project

1. **Switch to the *CCS Edit* perspective**
2. **Rebuild the project**
   - Menu *Project* → *Build Project*

CCS will automatically detect that the currently program being debugged has changed/rebuilt and ask if it should reload the file

3. **Select *Yes* and CCS will reload/reflash the program and run to main()**

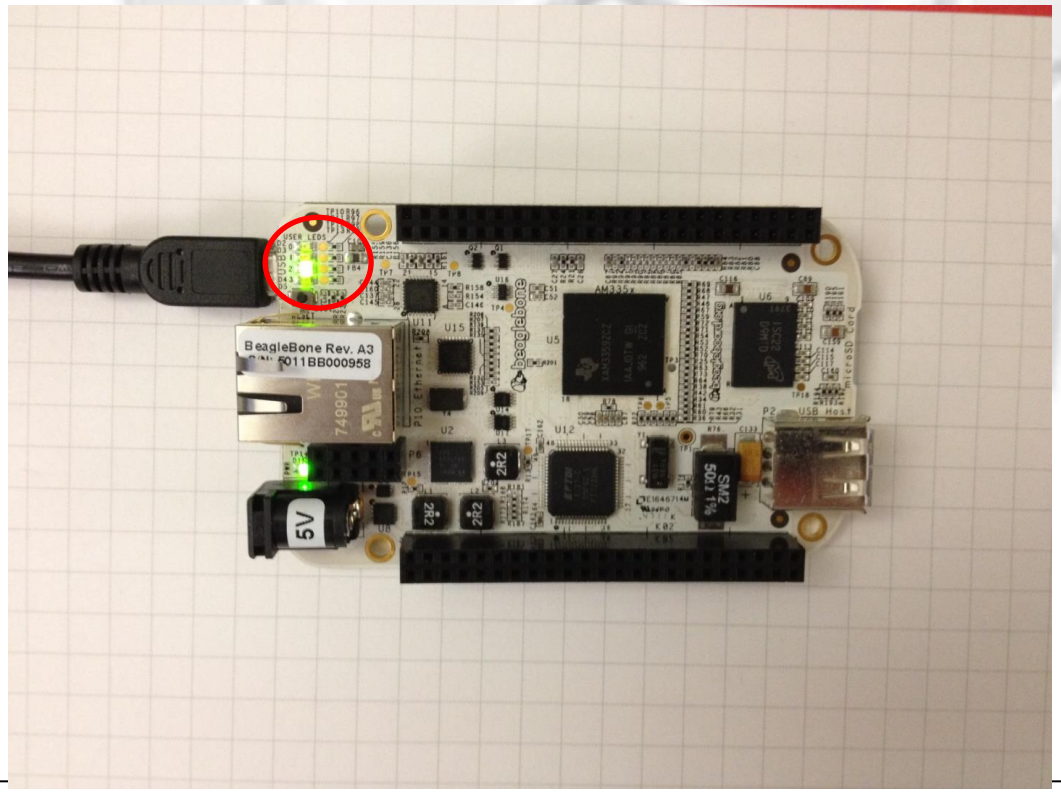




Blink USR2 LED

1. **Switch to the CCS Debug perspective**
2. **Press the *Run* button to run the program**

The USR2 LED on the BeagleBone should now be blinking at a much quicker rate
If not, remember to disable (not delete) the watchpoint set in the previous session! ☺



Debugging: Profile Clock – Part 2

1. **Pause execution of the program**
2. **Double-click on the clock icon in the status bar to reset the value to ‘0’**
3. **Enable the watchpoint for GPIO_SETDATAOUT**
4. **Click the *Run* button once**
5. **When the program stops, double-click again on the clock icon to reset its value to ‘0’**
6. **Click the *Run* button again**
   - Clock should now show ~101M cycles (1/2 of 203M)

Terminate the Debug Session

1. In the Breakpoint view, <u>delete</u> (not disable) the watchpoint, as it will not be used in the next lab – simply highlight it and press the <Delete> key.
2. Go to the Debug View
3. Click on the terminate button
4. This will kill the debugger and return you to the Edit perspective

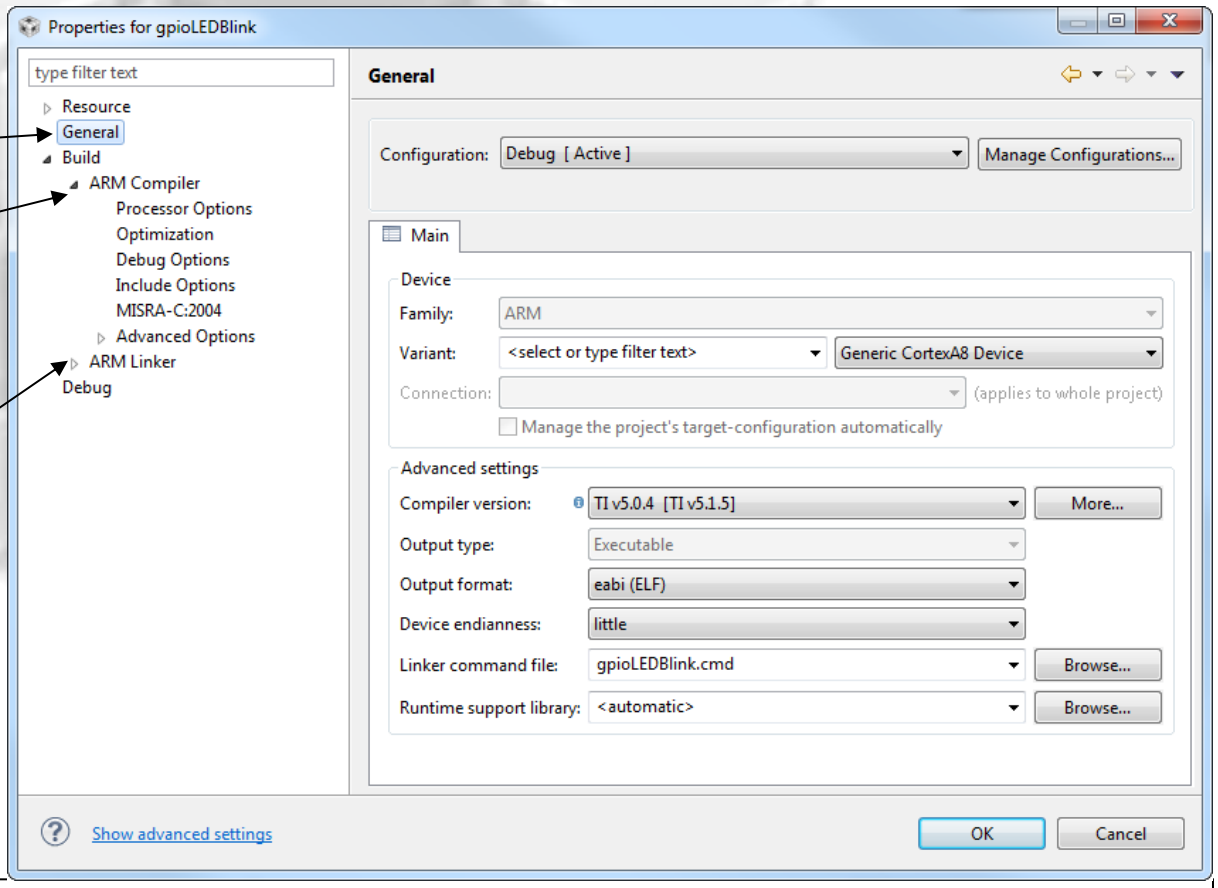
Changing Project Properties

1. Make sure you are in the *CCS Edit* perspective
2. Right click on the gpioLEDBlink project and select *Properties*

Device and high level settings

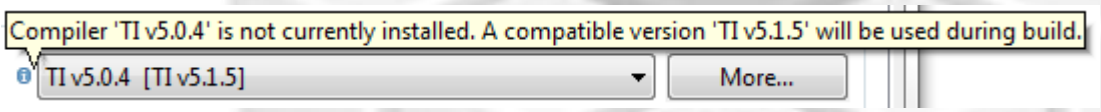
Compiler Options

Linker Options



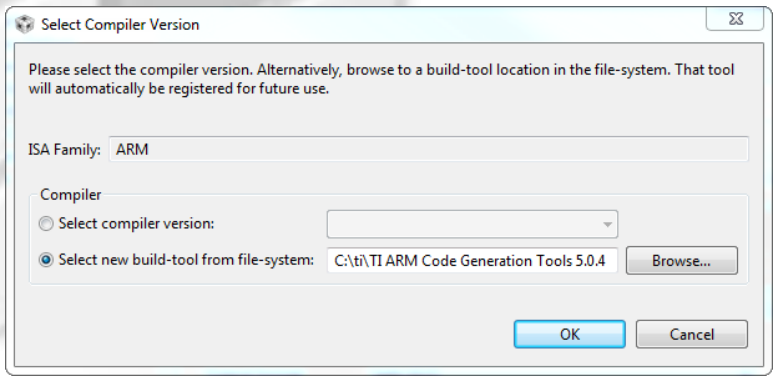
Changing the Compiler Version

1. **Click on** *General*
2. **Notice the small blue information icon next to Compiler version. Hovering over it shows the explanation**



3. **Click on the** *More…* **button beside the Compiler version** *TI v5.0.4*
4. **Check the option** *Select new build-tool from file-system*
5. **Browse to the location of the new compiler tools and click OK**

   C:\ti\TI ARM Code Generation Tools 5.0.4

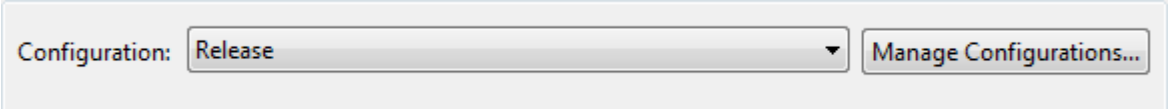


CCS will automatically find the compiler version, which matches with the original version used by the project

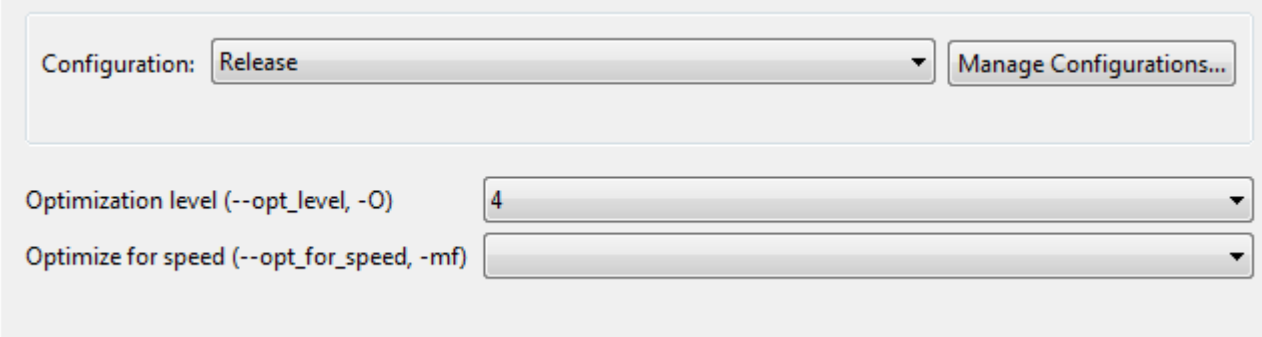
Changing Build Options

Build options are set per build configuration

1. **Change your Configuration to *Release***



2. **Change the optimization settings**
   - Go to the *Build* → *ARM Compiler* → *Optimization*
   - Change the optimization level to 4



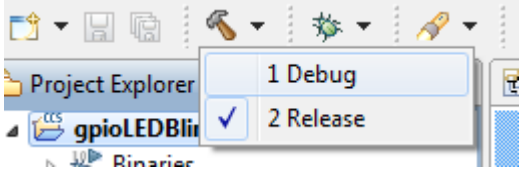
3. **Click OK**

Changing Build Options

1. **Change the active configuration to *Release***
   - Right click on the Project
   - Select *Build Configurations* → *Set Active* → *Release*
2. **Build the project by clicking the build button**
   - In the console view you will see that the *Release* configuration has been built

You can also change the configuration and build it by clicking on the arrow beside the build button and selecting the configuration you want to build

- Select *Release* and it will build this configuration
- The active configuration is indicated by the Checkmark



GPIO LED Blink Example: Exercise Summary

- **Congratulations! You finished all the lab steps for the first half.**
- **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
- **If you still have some time before the next half, please move to the optional steps of the lab:**
  - Debug symbols
  - Profile

Optional: Load the optimized code

1. After building the code, hit the “green bug” button

Despite the Release configuration does not use the option –g (symbolic debug), source code debug is still enabled. This is a new feature of TI compilers.

CCS Debug - gpioLEDBlink/gpioLEDBlink.c - Code Composer Studio

File Edit View Project Tools Scripts Run Window Help

Quick Access | CCS Edit | CCS Debug

Debug

gpioLEDBlink [Code Composer Studio - Device Debugging]
- Texas Instruments XDS100v2 USB Emulator_0/M3_wakeupSS_0 (Disconnected : Unknown)
- Texas Instruments XDS100v2 USB Emulator_0/CortxA8 (Suspended - SW Breakpoint)
  - main() at gpioLEDBlink.c:83 0x80000E70 (main does not contain frame information)
- Texas Instruments XDS100v2 USB Emulator_0/PRU_0 (Disconnected : Unknown)
- Texas Instruments XDS100v2 USB Emulator_0/PRU_1 (Disconnected : Unknown)

Variables | Expressions | Registers | Breakpoints

Identity	Name	Condition	Count	Action

BeagleBone.cc... | gpioLEDBlink.c | gpio_v2.c | Compare gpioL...

```
78 ****************************************************************
79 /*
80 ** The main function. Application starts here.
81 */
82 int main()
83 {
84     /* Enabling functional clocks for GPIO1 instance. */
85     GPIO1ModuleClkConfig();
86
87     /* Selecting GPIO1[23] pin for use. */
88     GPIO1Pin23PinMuxSetup();
89
90     /* Enabling the GPIO module. */
91     GPIOModuleEnable(GPIO_INSTANCE_ADDRESS);
92
93     /* Resetting the GPIO module. */
94     GPIOModuleReset(GPIO_INSTANCE_ADDRESS);
```

Disassembly | main

```
          main():
80000e70:   E92D4038 STMFD    R13!, {R3, R4, R5, R14}
80000e74:   EBFFFF39 BL       GPIO1ModuleClkConfig
80000e78:   EBFFFF34 BL       GPIO1Pin23PinMuxSetup
80000e7c:   E59F4054 LDR      R4, $C$CON1
80000e80:   E1A00004 MOV      R0, R4
80000e84:   EBFFFE24 BL       GPIOModuleEnable
80000e88:   E1A00004 MOV      R0, R4
80000e8c:   EBFFFE1B BL       GPIOModuleReset
80000e90:   E3A02000 MOV      R2, #0
80000e94:   E3A01017 MOV      R1, #23
80000e98:   E1A00004 MOV      R0, R4
100           while(1)
80000e9c:   EBFFFEA2 BL       GPIODirModeSet
101           {
          $C$L3:
80000ea0:   E3A01017 MOV      R1, #23
```

Console

GPIO LED Blink Example: Optional Exercise Summary

- **Congratulations! You finished all the optional lab steps.**
- **At this point you experimented the following concepts:**
  - Debug symbols
  - Profile

⚠ **Stop at this time and wait for the next half of the presentation.**

BLANK

UART Echo Example: Exercise Summary

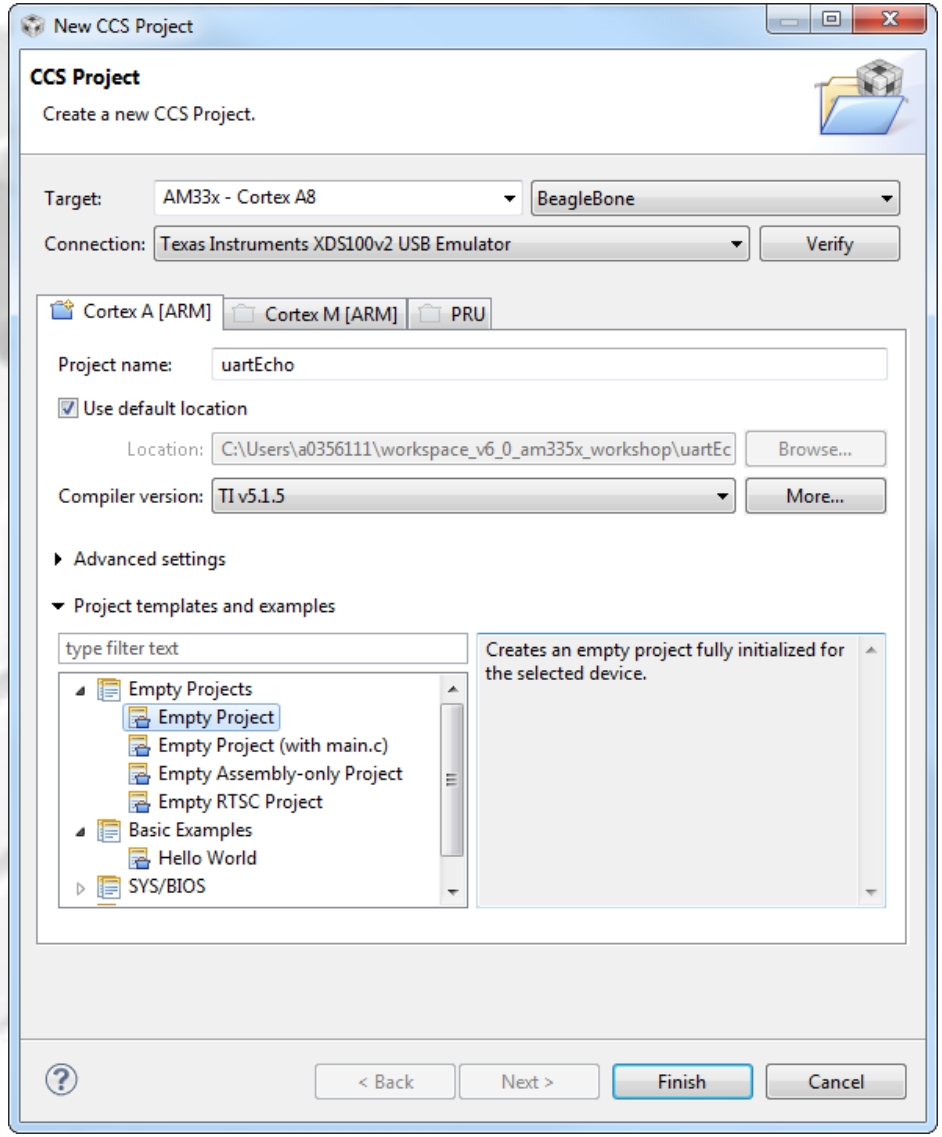
- **Key Objectives**
  - Create a new portable project based on the UART Echo example
  - Create workspace level variables for the project
  - Link files to the project using variables
  - Configure build properties using variables
  - Validate project by building, loading and running the program
- **Tools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

Create a New Project

1. **Launch the *New CCS Project* Wizard**
   - Go to menu File → *New* → *CCS Project*
2. **Fill in the fields as shown in the right**
   - *Target:* AM33x – Cortex A8 → BeagleBone
   - *Connection:* Texas Instruments XDS100v2 USB Emulator
   - *Tab:* Cortex A [ARM]
   - *Project name:* uartEcho
   - *Compiler version:* TI v5.1.5
3. **Click *Finish* when done**

Generated project will appear in the Project Explorer view

4. **Remove the generated file <AM335x.cmd> from the project**



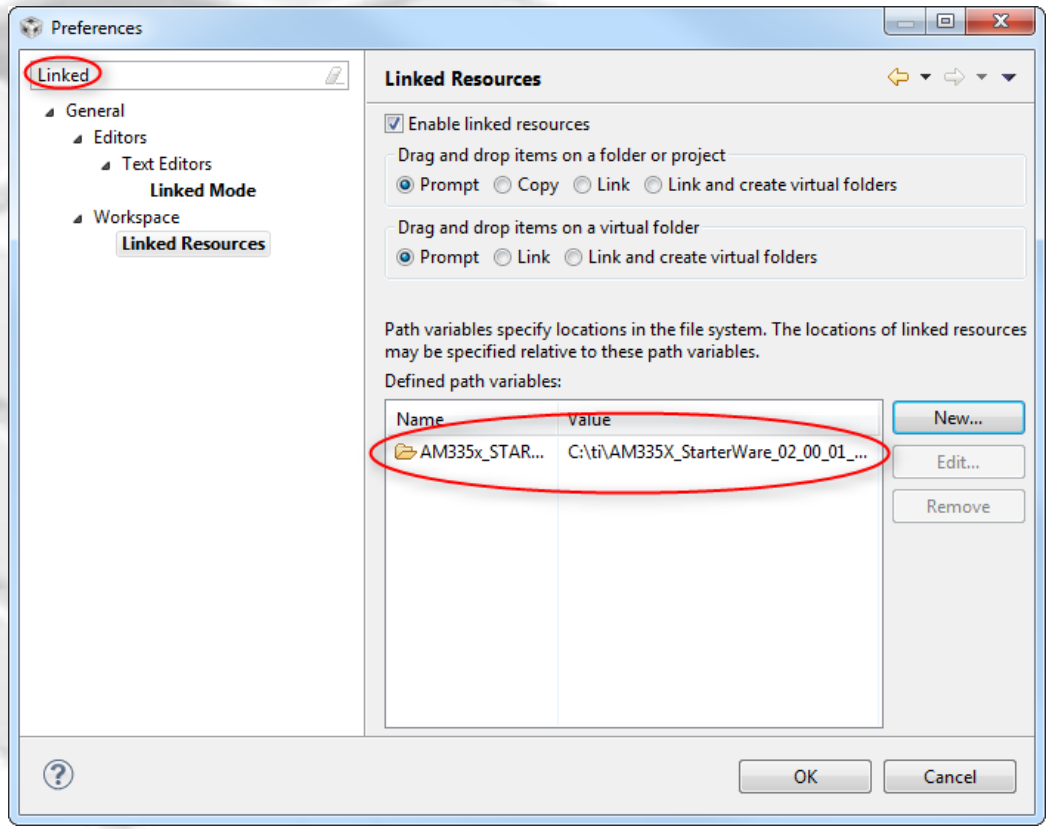
Create a Linked Resource Path Variable

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

1. **Open the workspace preferences**
   - Menu *Window* → *Preferences*
2. **Go to the *Linked Resources* preferences**
   - Type 'Linked' in the filter field to make it easier to find
3. **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_01_01

4. **Hit *OK* when done**



Link Source Files to Project

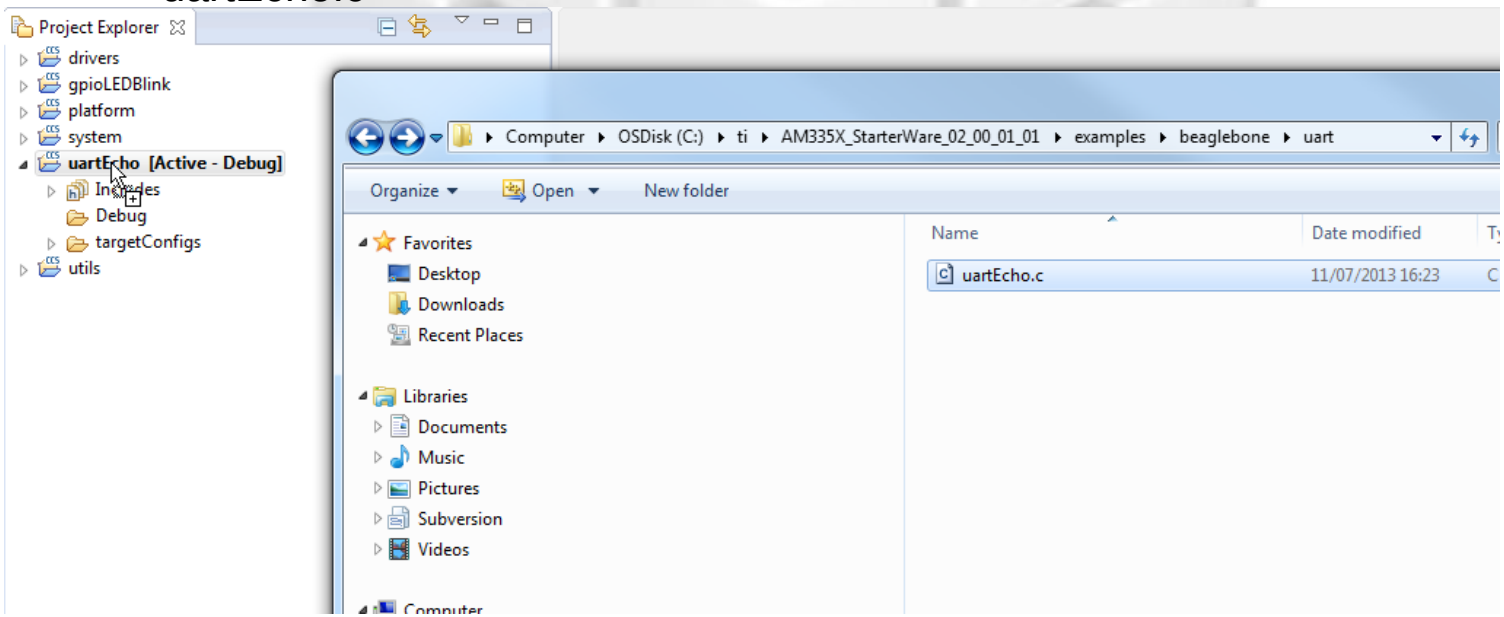
Here we will link the source file relative to the Linked Resource Path Variable previously created

1. **Open Windows Explorer and browse to:**

   C:\ti\AM335X_StarterWare_02_00_01_01\examples\beaglebone\uart

2. **Drag and drop the following file into the new project in the CCS Project Explorer view**

   – <uartEcho.c>

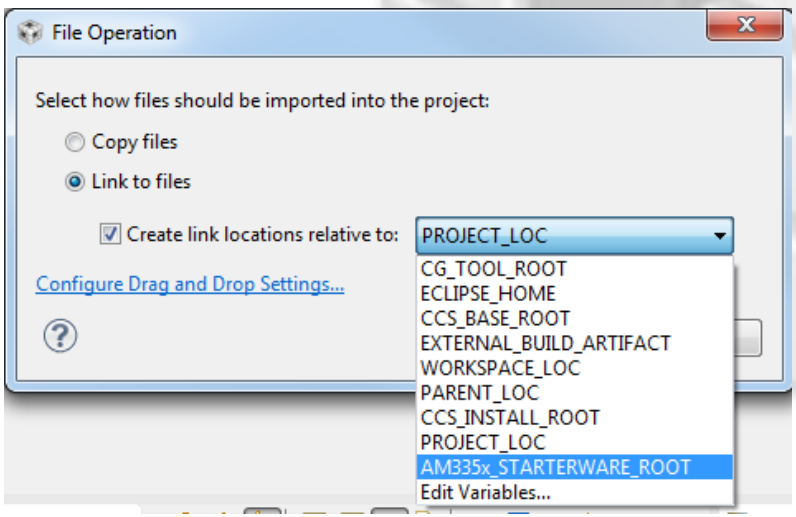


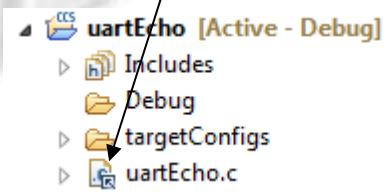
Link Source Files to Project

A dialog will appear asking if you wish to Copy or Link the files:

1. **Select *Link to files***
2. **Select *Create link locations relative to:***
   - Use the new Linked Resource variable we created (**AM335x_STARTERWARE_ROOT**)
3. **Hit *OK***

Files will now appear in the Project Explorer with the 'link' icon

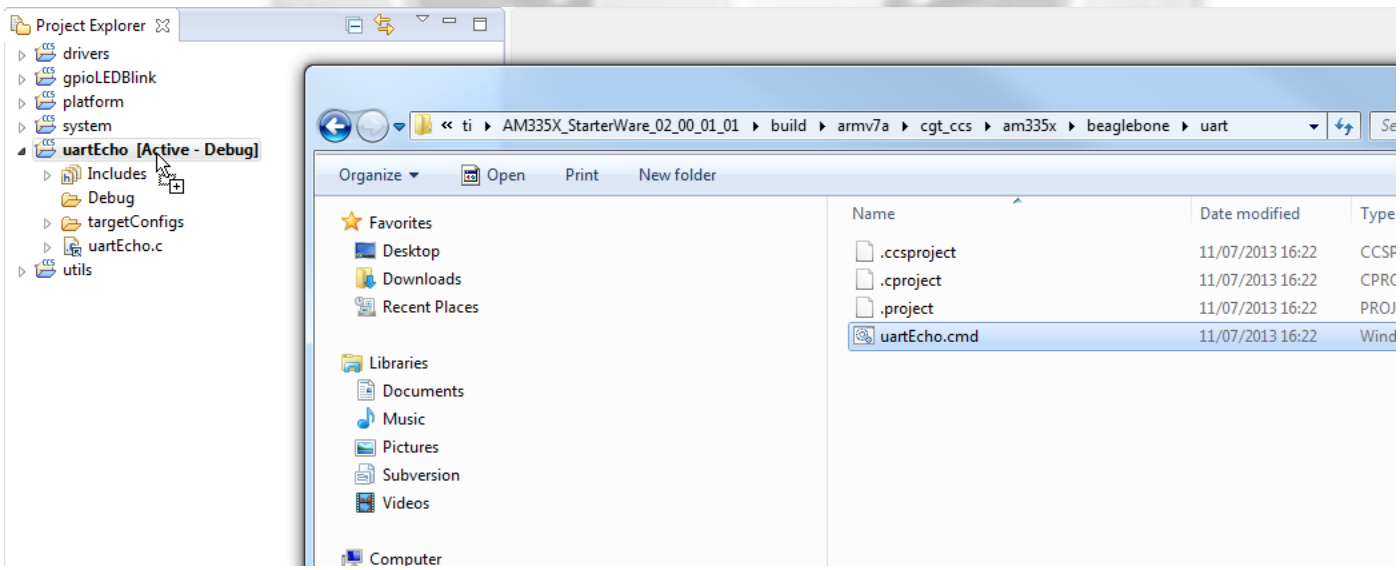




Add Files to Project

Here we will add the linker command file to the project. Ideally this file should also be "linked" but just to show the other way, we will "copy" the file instead

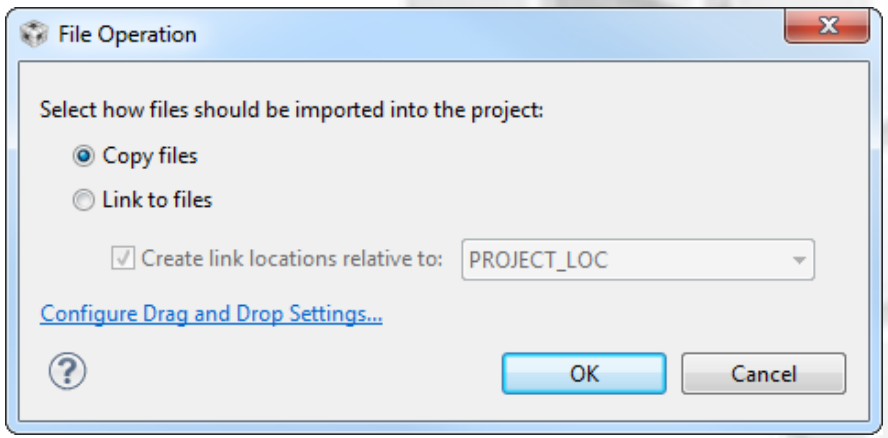
1. **Open Windows Explorer and browse to:**
   C:\ti\AM335X_StarterWare_02_00_01_01\build\armv7a\cgt_ccs\am335x\beaglebone\uart
2. **Drag and drop the following file into the new project in the CCS Project Explorer view**
   – <uartEcho.cmd>

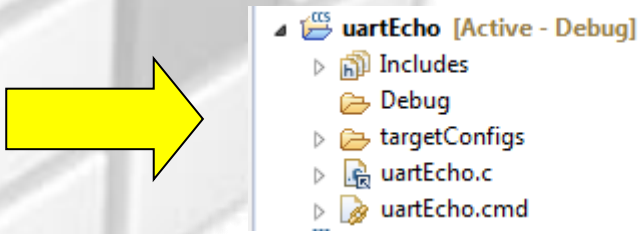


Add Files to Project

A dialog will appear asking if you wish to Copy or Link the files:

1. Select *Copy files*
2. Hit *OK*

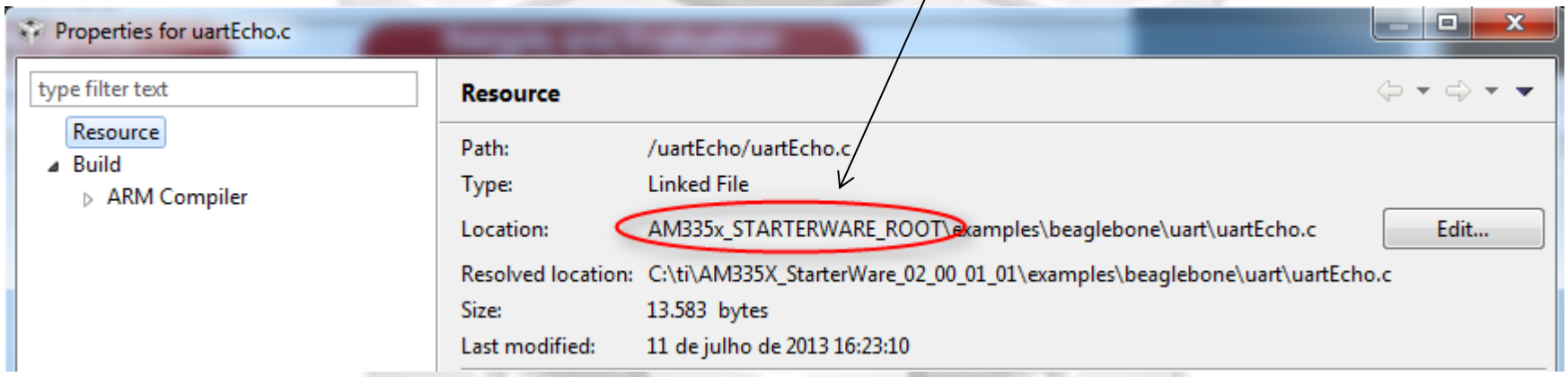




Link Files to Project

1. **Right-click on the C source file and check the *Properties***

See how the *Location* parameter references the Linked Resource Variable



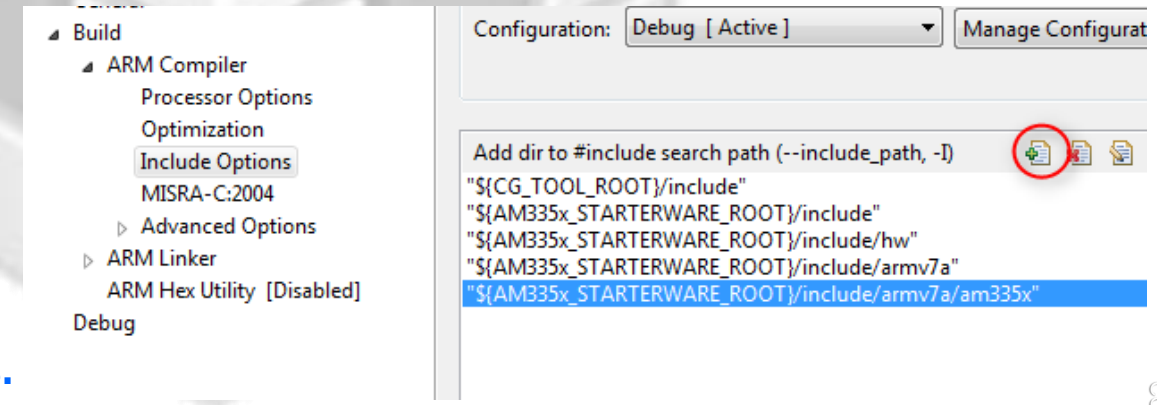
Modifying Project Properties

Here we are adding paths to include files using the Build Variable (created when the Link Resource Path variable was set)

1. **Right-click on the project and select *Properties***
2. **In the compiler *Include Options*, click on the "Add" button to 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
3. **Click *OK***

'${<BUILD VARIABLE>}' is the syntax to use a Build Variable in the project properties

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



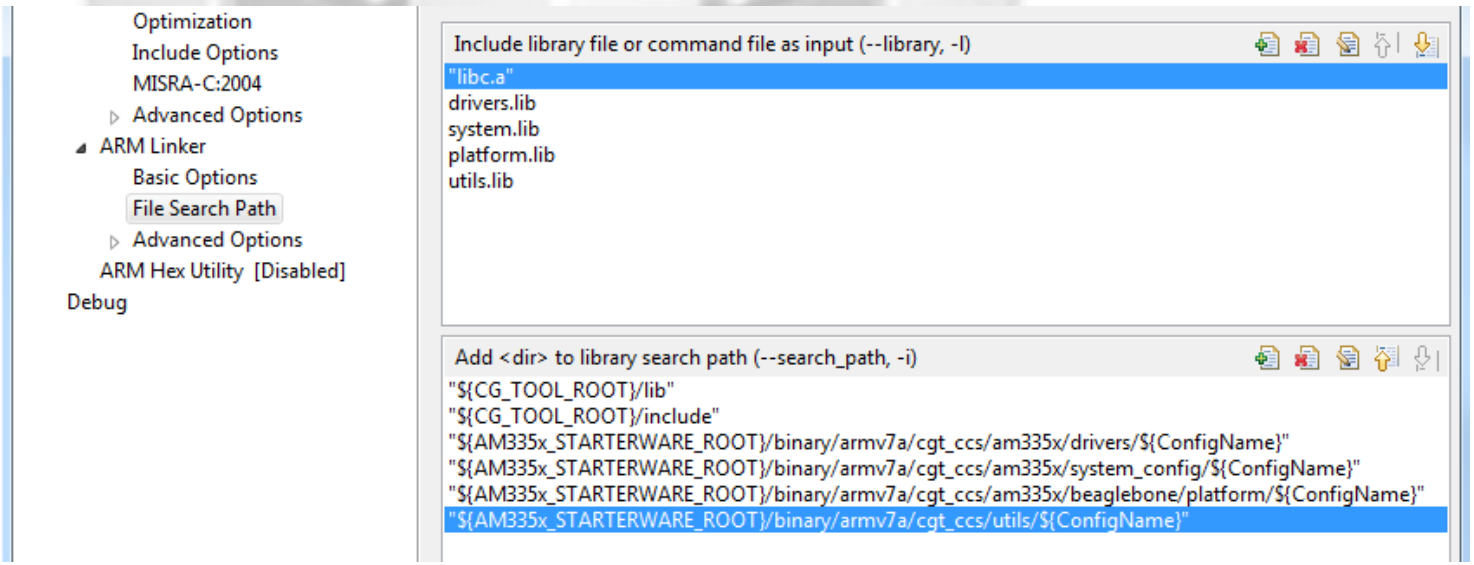
Modifying Project Properties

Here we are adding paths to libraries using the Build Variable

1. **Right-click on the project and select *Properties***

2. **In the Linker *File Search Path*, add the following entries under *--search_path* option:**
   - ${AM335x_STARTERWARE_ROOT}/binary/armv7a/cgt_ccs/am335x/drivers/${ConfigName}
   - ${AM335x_STARTERWARE_ROOT}/binary/armv7a/cgt_ccs/am335x/system_config/${ConfigName}
   - ${AM335x_STARTERWARE_ROOT}/binary/armv7a/cgt_ccs/am335x/beaglebone/platform/${ConfigName}
   - ${AM335x_STARTERWARE_ROOT}/binary/armv7a/cgt_ccs/utils/${ConfigName}

3. **In the Linker *File Search Path*, add the following entries under *--library* option:**
   - drivers.lib
   - system.lib
   - platform.lib
   - utils.lib



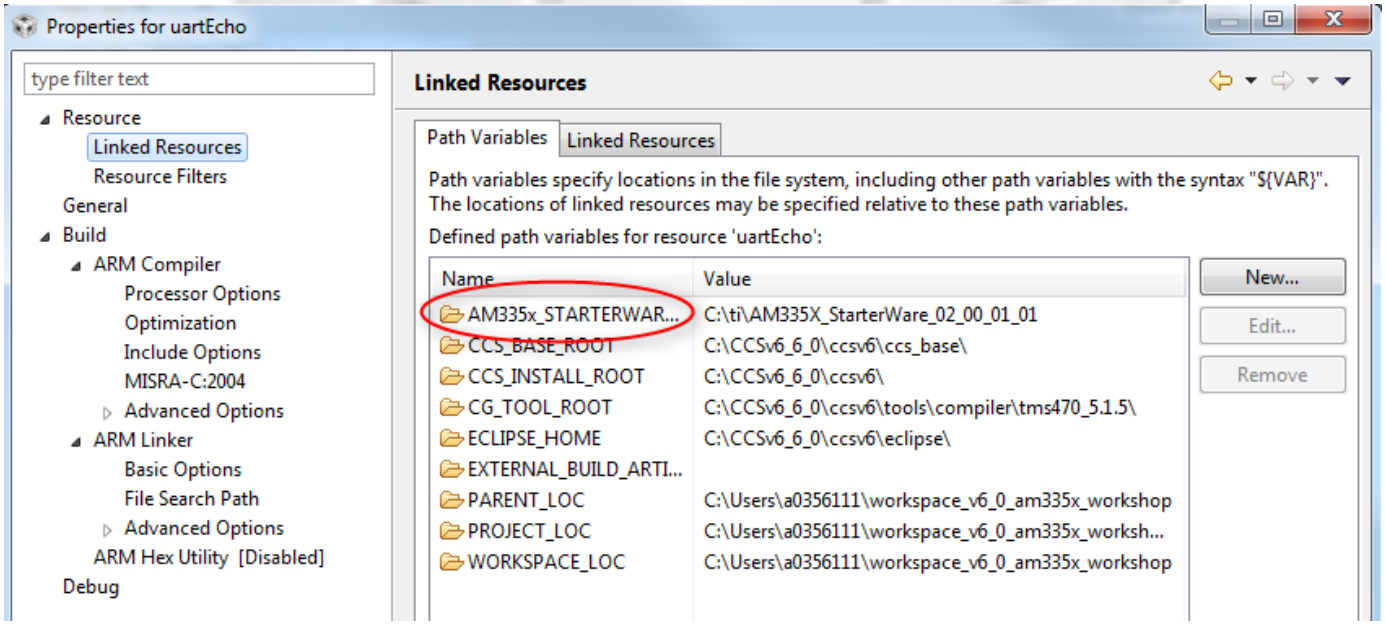
Project Properties

1. **Go to *Resource* → *Linked Resources* to see all the Linked Resource Path Variables that are available to the project**

ⓘ This will show all variables created at the project level and workspace level

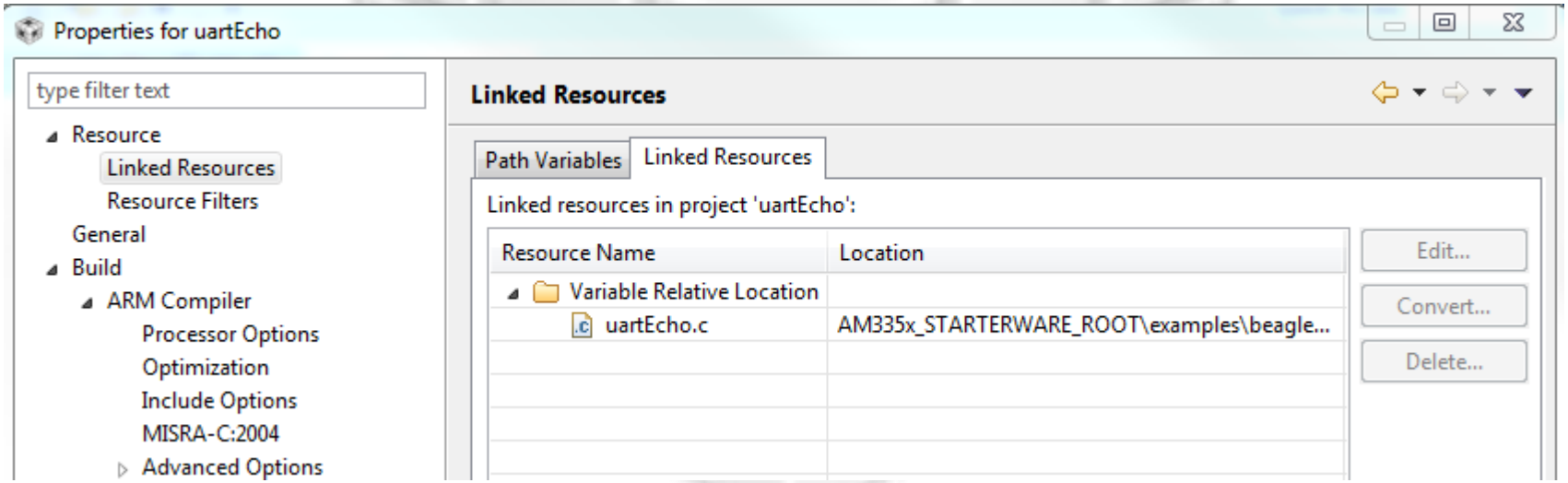
2. **Check if the workspace level Linked Resource Path Variable that was created appears in the list**

ⓘ Variables may be edited here but changes will only be recorded at the project level (stored in the project files)



Project Properties

- **The *Linked Resources* tab will show all the files that have been linked to the project**
  - It will sort them by files linked with a variable and files linked with an absolute path
- **Links can be modified here with the *Edit…* button**
- **Links can be converted to use an absolute path with the *Convert…* button**



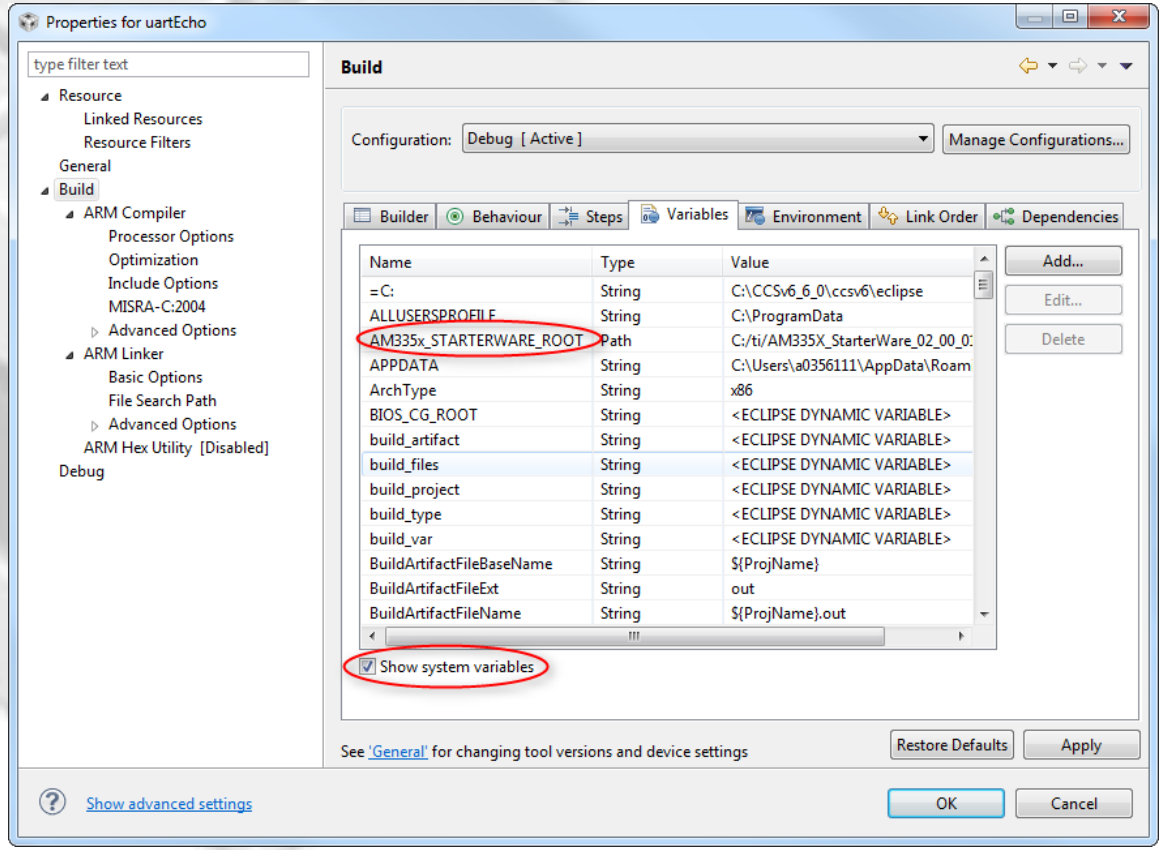
Project Properties

1. **Go to *Build* → tab *Variables* to see all the Build Variables that are available to the project**

   Only project level variables will be listed by default

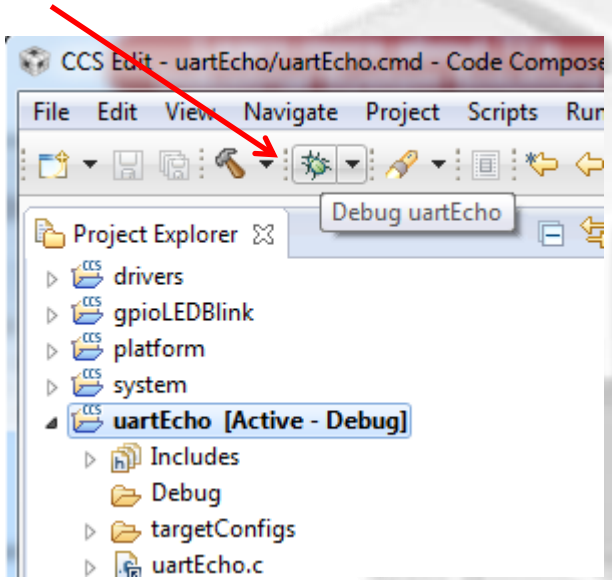
2. **Enable the *Show system variables* checkbox to see variables set at the workspace and system level**

   See how the workspace level Build Variable that was created appears in the list



Build and Load the Program

1. Make sure the ‘uartEcho’ project is in focus. Then use the *Debug* button to build and load the code



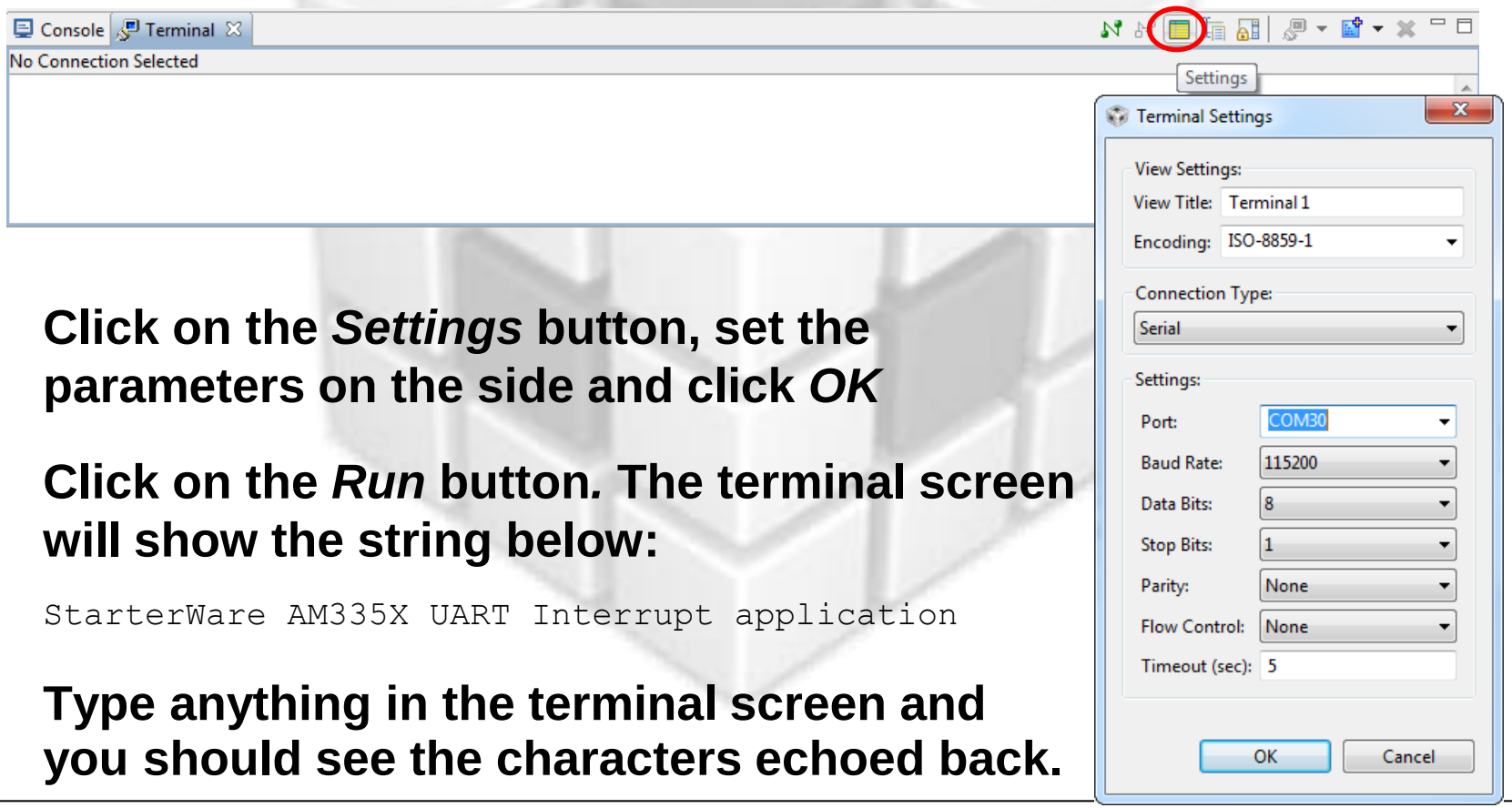
2. When the Debug Session is launching, CCS asks which cores to load. De-select **M3_wakeupSS_0**

This happens because BeagleBone has two cores compatible with the project (ARM cores)

Seeing the output on the terminal plug-in

1. **Go to menu *Window* → *Show View* → *Other***

2. **Type 'Terminal' in the text box, select the *Terminal* entry and click *OK***



3. **Click on the *Settings* button, set the parameters on the side and click *OK***

4. **Click on the *Run* button. The terminal screen will show the string below:**

```
StarterWare AM335X UART Interrupt application
```

5. **Type anything in the terminal screen and you should see the characters echoed back.**

UART Echo Example: Exercise Summary

- **Congratulations! You finished all the labs.**
- **At this point you experimented the following concepts:**
  - Portable Projects
  - Linked resources
  - Linked resource path variables
  - Build variables
- **Please don't forget to answer the evaluation form before leaving; it is very important for us to improve this session.**
- **Also, make sure to save any files and projects you want to take home.**
- **Thank you, enjoy the rest of your day and have a safe trip back.**