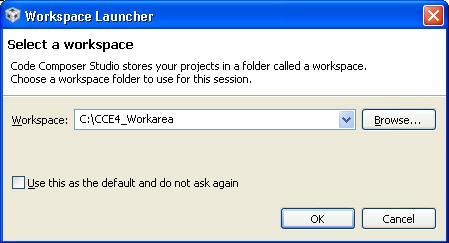
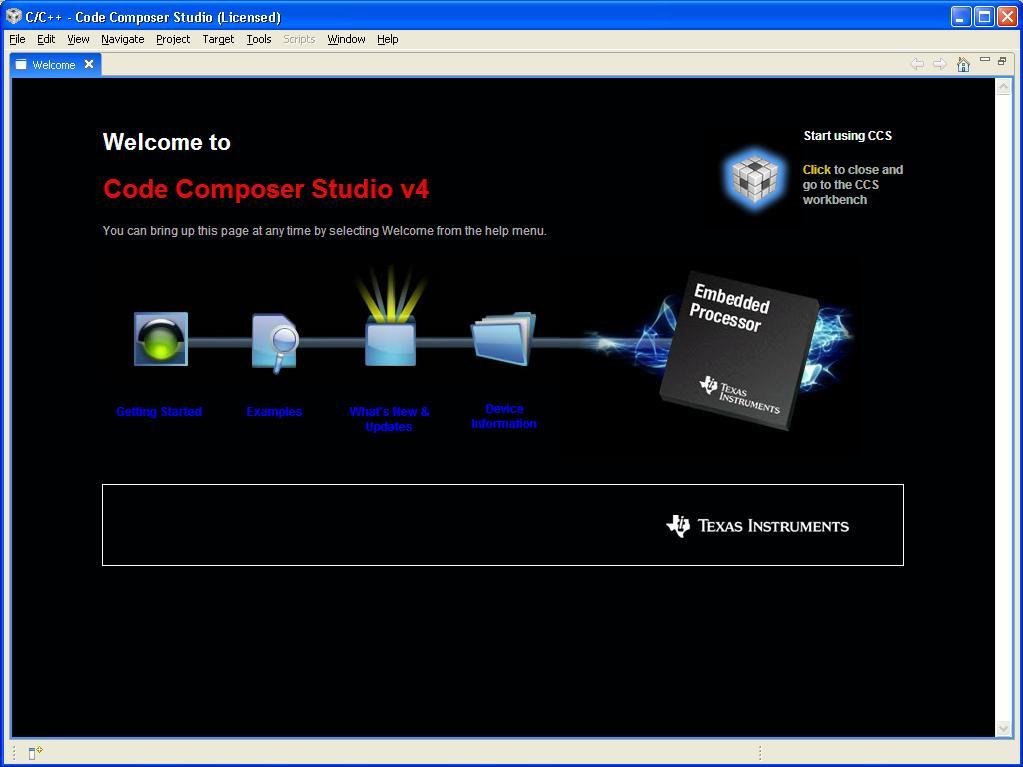
Getting started with Code Composer V4, JTAG, and the Beagleboard

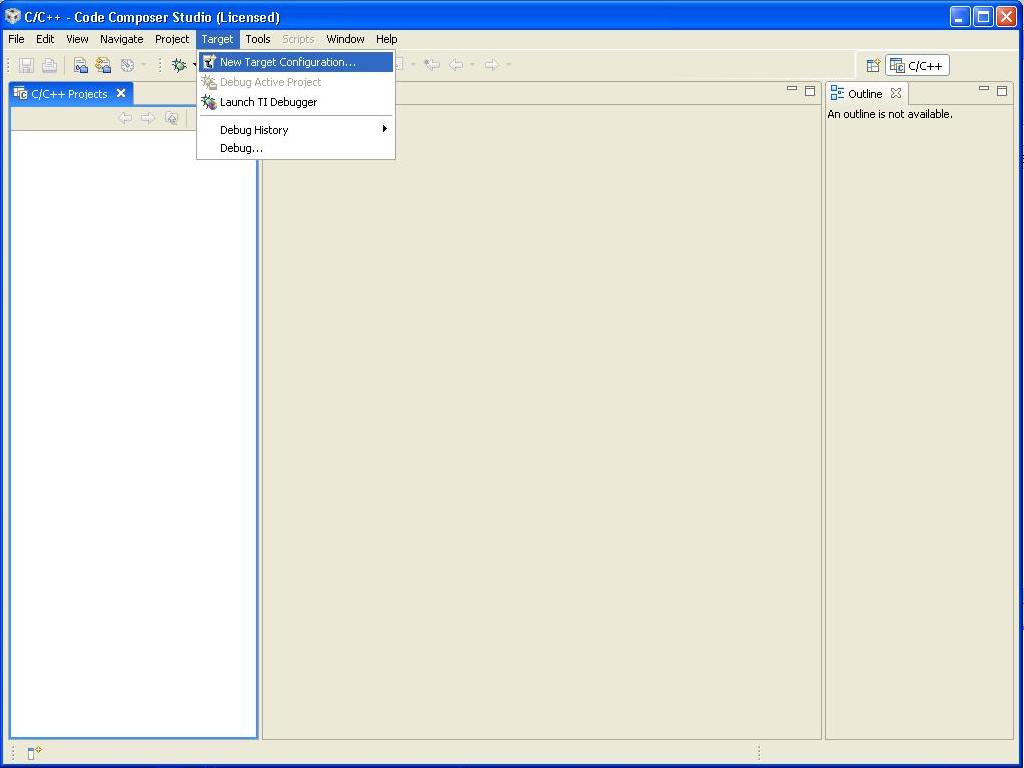
This guide was written using CCS V4.1



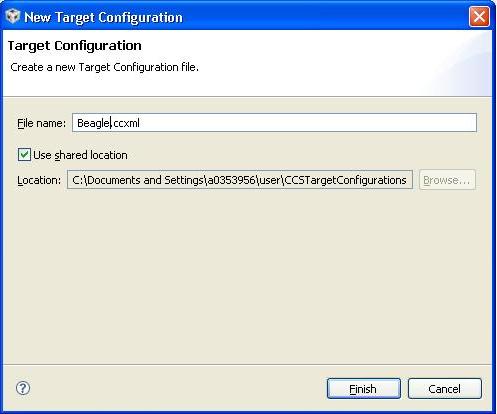
1. Start CCS V4.   
   You will be presented with the Workspace Launcher. For this example we’ll use a work area called “CCE4\_Workarea”



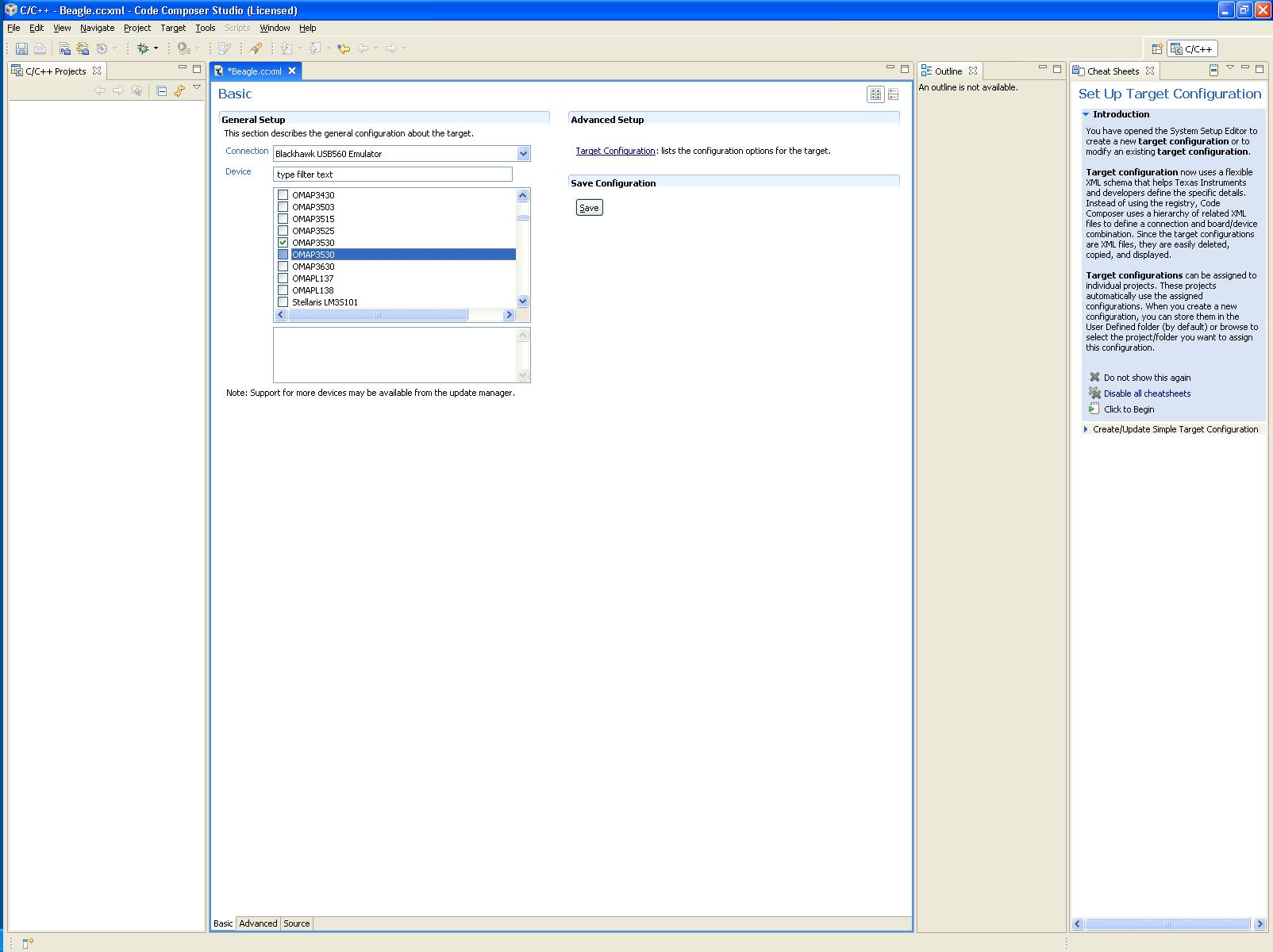
1. If this is the first time this work area has been used you will be presented with the “Welcome to” screen. Simply click on the cube in the top right of the screen to continue



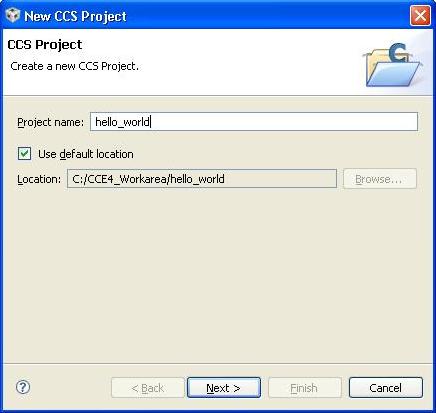
1. The main CCS screen should now appear. The first thing to be done is create a Target Configuration to detail the target device, GEL files, and JTAG Emulator. Click Target-> New Target Configuration.



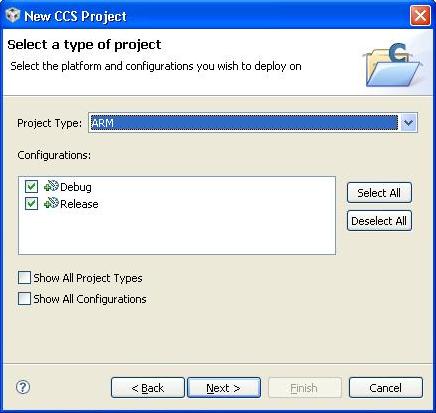
1. Enter a descriptive name for the configuration. If you have multiple JTAG emulator and multiple boards I suggest you include those in the name to help easily identify and select configurations in the future. For this example we will use the name “Beagle.ccxml”.



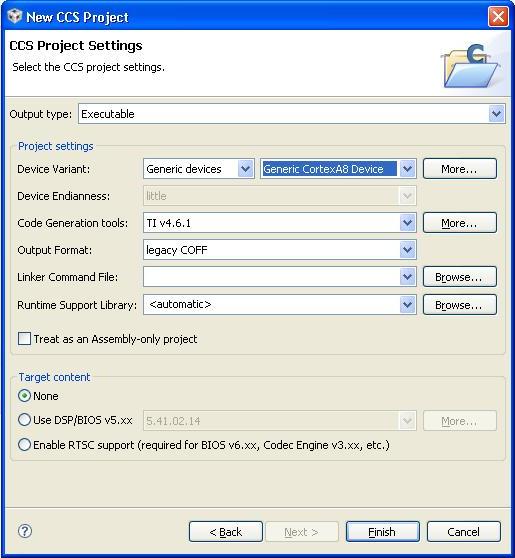
1. Select the JTAG emulator you are using from “Connection” drop down list and then select the device from the “Device list”. Here we’ll use the “Blackhawk USB560 Emulator” for the Connection and “OMAP3530” for the device. Now click Save! You can click on the “Advanced” tab at the bottom if you’d like to explore the setup and see which GEL files are used.



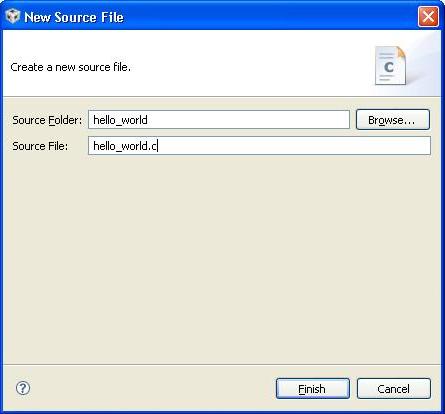
1. Now create a project. Click File->New->CCS Project. The New CCS Project dialogue box should popup. Enter “hello\_world“ as the project name and click next.



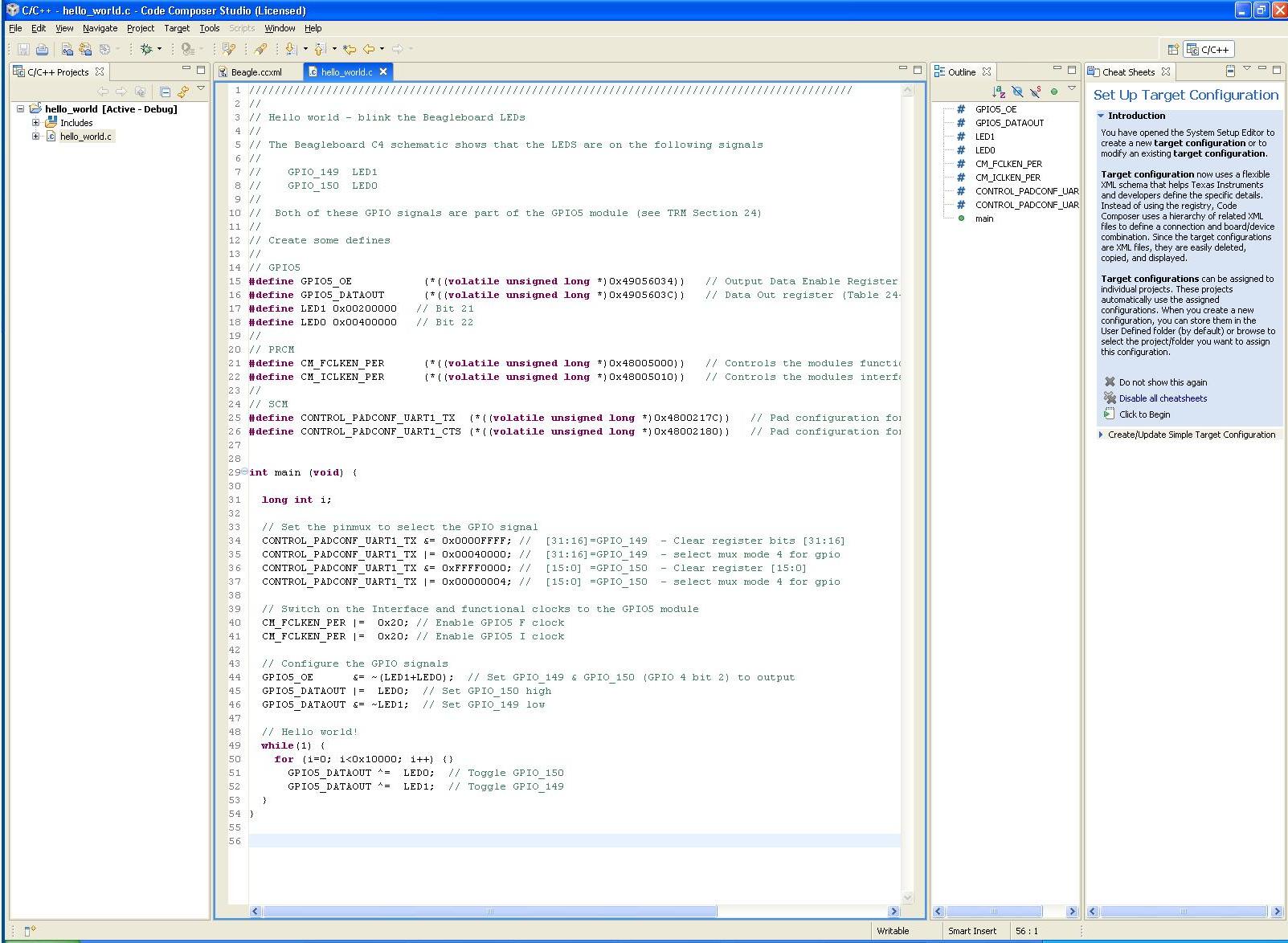
1. The Project Type should be set to “ARM”. Leave everything else default and click next. For the “Additional Project Settings” just click next.



1. You should now see the “CCS Project Settings”. Change “Device Variant“ to “Generic devices” & “Generic CortexA8 Device”. Leave everything default. We’ll create a linker file a little later. Click Finish.



1. Create the source code. Right click on the project name (hello\_world) then select New->Source File. Enter “hello\_world.c” as the Source file and click Finish



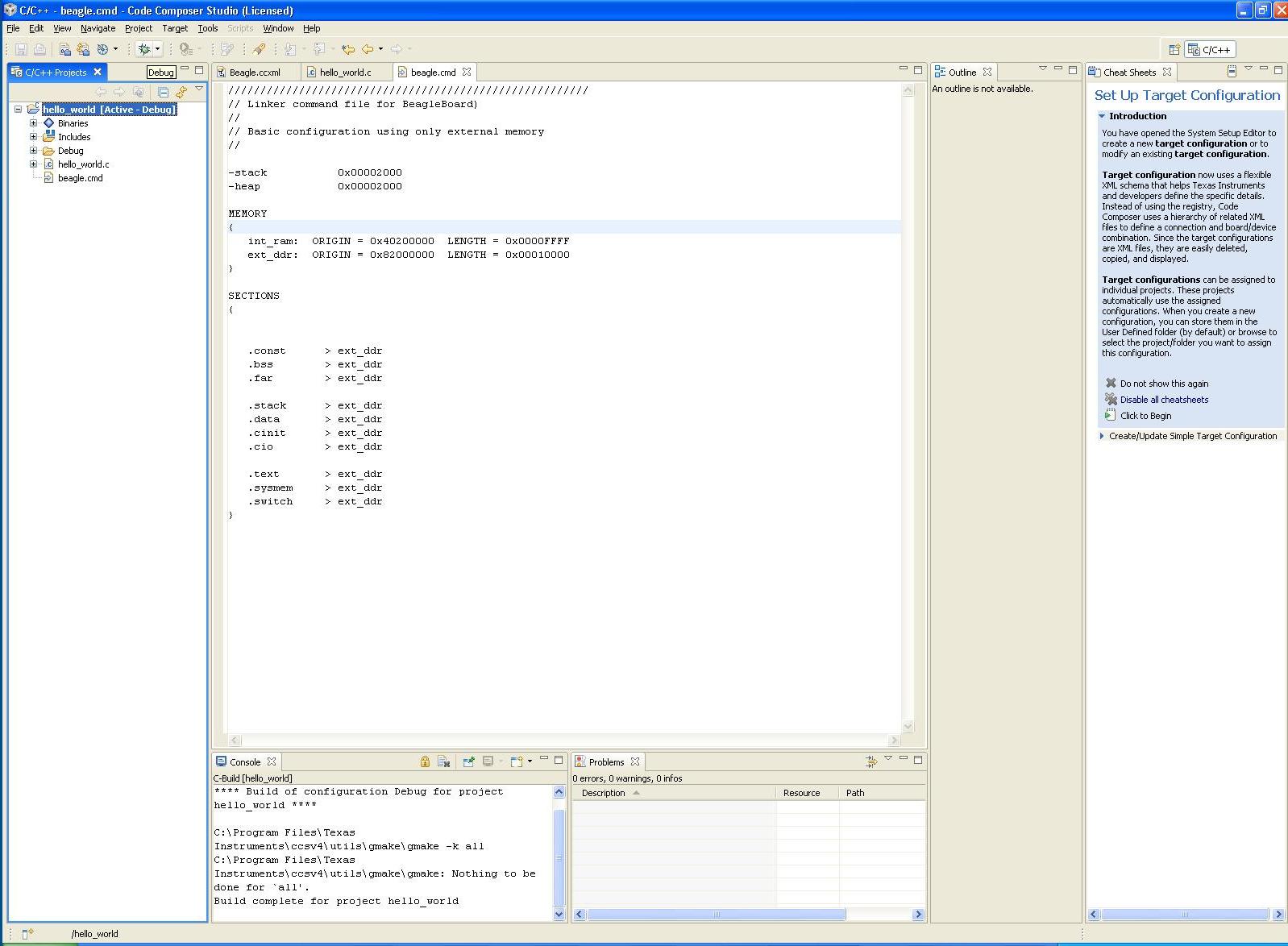
1. A new tab should open called “hello\_world.c” This is the editor window. Paste the source code to the window and ctrl-s to save.



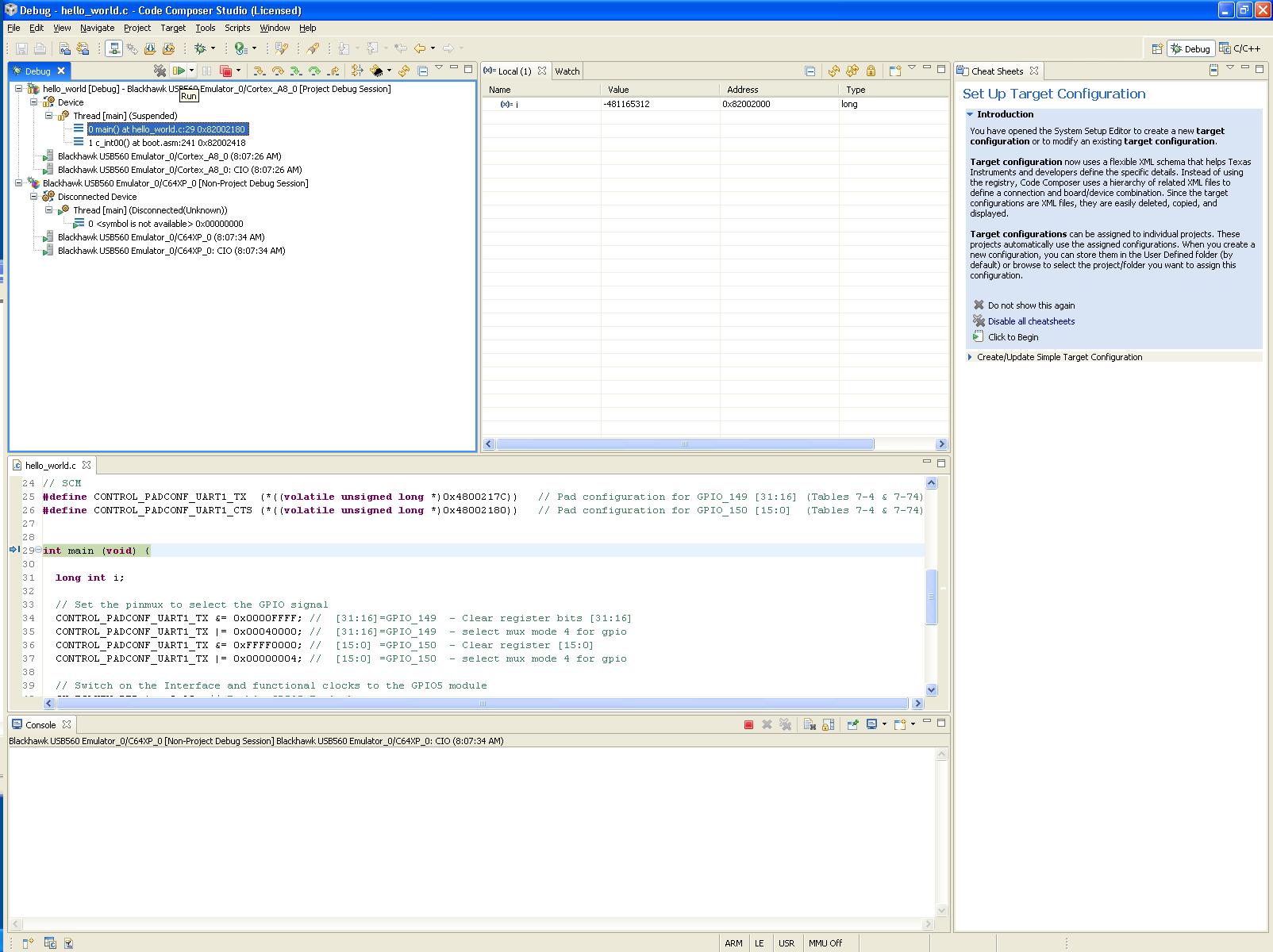
1. Create the Linker command file. Right click on the project name (hello\_world) then select New->File. Enter “beagle.cmd” as the File name and click Finish



1. A new tab should open called “beagle.cmd”. This is the editor window. Paste the command file to the window and ctrl-s to save.



1. To compile the program right click on the project name (hello\_world) then select Build Project. Two new windows should open up at the bottom called Console and Problems. If everything went ok there should be no errors and the last line of the console should read “Build complete for project hello\_world”. Make sure the JTAG emulator is connected and that the Beagleboard is powered. Download the code to the Beagleboard by clicking on the green bug located on the top icon bar (circled).



1. The program should download and be halted at the start of “main”. Click on the green play button to run the program

Hello\_world.c:

//////////////////////////////////////////////////////////////////////////////////////////////

//

// Hello world - blink the Beagleboard LEDs

//

// The Beagleboard C4 schematic shows that the LEDS are on the following signals

//

// GPIO\_149 LED1

// GPIO\_150 LED0

//

// Both of these GPIO signals are part of the GPIO5 module (see TRM Section 24)

//

// Create some defines

//

// GPIO5

**#define** GPIO5\_OE (\*((**volatile** **unsigned** **long** \*)0x49056034)) // Output Data Enable Register (Table 24-27)

**#define** GPIO5\_DATAOUT (\*((**volatile** **unsigned** **long** \*)0x4905603C)) // Data Out register (Table 24-31)

**#define** LED1 0x00200000 // Bit 21

**#define** LED0 0x00400000 // Bit 22

//

// PRCM

**#define** CM\_FCLKEN\_PER (\*((**volatile** **unsigned** **long** \*)0x48005000)) // Controls the modules functional clock activity. (Table 4-237)

**#define** CM\_ICLKEN\_PER (\*((**volatile** **unsigned** **long** \*)0x48005010)) // Controls the modules interface clock activity (Table 4-239)

//

// SCM

**#define** CONTROL\_PADCONF\_UART1\_TX (\*((**volatile** **unsigned** **long** \*)0x4800217C)) // Pad configuration for GPIO\_149 [31:16] (Tables 7-4 & 7-74)

**#define** CONTROL\_PADCONF\_UART1\_CTS (\*((**volatile** **unsigned** **long** \*)0x48002180)) // Pad configuration for GPIO\_150 [15:0] (Tables 7-4 & 7-74)

**int** main (**void**) {

**long** **int** i;

// Set the pinmux to select the GPIO signal

CONTROL\_PADCONF\_UART1\_TX &= 0x0000FFFF; // [31:16]=GPIO\_149 - Clear register bits [31:16]

CONTROL\_PADCONF\_UART1\_TX |= 0x00040000; // [31:16]=GPIO\_149 - select mux mode 4 for gpio

CONTROL\_PADCONF\_UART1\_TX &= 0xFFFF0000; // [15:0] =GPIO\_150 - Clear register [15:0]

CONTROL\_PADCONF\_UART1\_TX |= 0x00000004; // [15:0] =GPIO\_150 - select mux mode 4 for gpio

// Switch on the Interface and functional clocks to the GPIO5 module

CM\_FCLKEN\_PER |= 0x20; // Enable GPIO5 F clock

CM\_FCLKEN\_PER |= 0x20; // Enable GPIO5 I clock

// Configure the GPIO signals

GPIO5\_OE &= ~(LED1+LED0); // Set GPIO\_149 & GPIO\_150 (GPIO 4 bit 2) to output

GPIO5\_DATAOUT |= LED0; // Set GPIO\_150 high

GPIO5\_DATAOUT &= ~LED1; // Set GPIO\_149 low

// Hello world!

**while**(1) {

**for** (i=0; i<0x10000; i++) {}

GPIO5\_DATAOUT ^= LED0; // Toggle GPIO\_150

GPIO5\_DATAOUT ^= LED1; // Toggle GPIO\_149

}

}

Linker command file:

////////////////////////////////////////////////////////

// Linker command file for BeagleBoard)

//

// Basic configuration using only external memory

//

-stack 0x00002000

-heap 0x00002000

MEMORY

{

int\_ram: ORIGIN = 0x40200000 LENGTH = 0x0000FFFF

ext\_ddr: ORIGIN = 0x82000000 LENGTH = 0x00010000

}

SECTIONS

{

.const > ext\_ddr

.bss > ext\_ddr

.far > ext\_ddr

.stack > ext\_ddr

.data > ext\_ddr

.cinit > ext\_ddr

.cio > ext\_ddr

.text > ext\_ddr

.sysmem > ext\_ddr

.switch > ext\_ddr

}