Sitara Linux Training: Tuning the DDR3 Timings on BeagleBoneBlack

From Texas Instruments Wiki

Return to the **Sitara Linux Training List**

Contents

Introduction

This lab is going to walk you through the process of determining and configuring the DDR3 values for a custom board. For the purposes of this lab we will be using the BeagleBone Black (specifically rev A5). Note that the BeagleBone Black JTAG header is not populated out of the box, therefore you will need to add the 20-pin CTI header on the under-side of the board.

Determine Basic Preliminary Timings

Description

This section discusses how to determine the preliminary DDR3 timings for the EMIF on the AM335x. This will be done by entering data from the memory datasheet into a spreadsheet as well as entering some board-specific information into a second spreadsheet. The resulting output data will be placed into a GEL file which CCS will use to configure the EMIF registers.

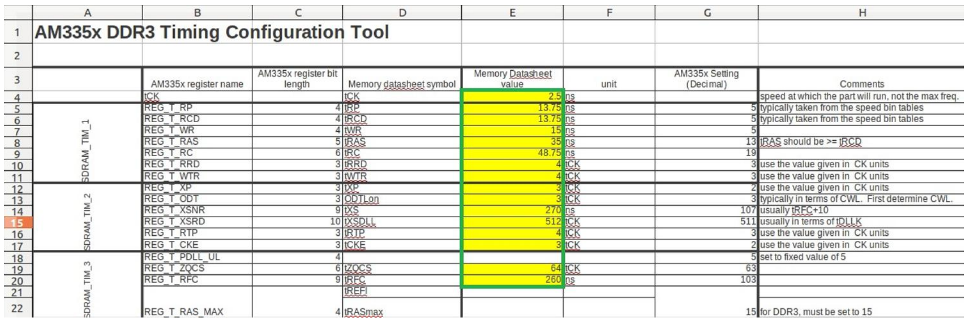
Prerequisites

- The memory datasheet(s) for the DDR3 chip(s) on the board
- RatioSeed Tool
  - From http://processors.wiki.ti.com/index.php/AM335x_DDR_PHY_register_configuration_for_DDR3
- DDR3 Timing Configuration Tool
  - From http://processors.wiki.ti.com/index.php/AM335x_EMIF_Configuration_tips
- This sample BeagleBoneBlack GEL file: File:BeagleBlack 400Mhz 4GbDDR.gel.tar.gz

Lab Steps

Open the DDR3 Timing Configuration Tool spreadsheet and the memory datasheet

1. Inside the spreadsheet there are a number of different fields which must be populated per the memory datasheet
   - Note that every value must be filled in as the memory datasheet specifies regardless of what frequency the memory is actually operating
   - The **one exception** is the operating frequency (tck) itself - we will use **400MHz instead of the max 800MHz** for the DDR3-1600 memory used on the BeagleBoneBlack. This is because the AM335x EMIF cannot operate at 800MHz. Refer to the AM335x TRM in the EMIF chapter for more information on the EMIF timing requirements

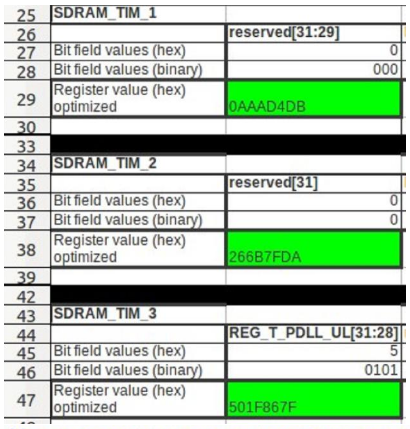


	A	B	C	D	E	F	G	H
1	AM335x DDR3 Timing Configuration Tool							
2								
3		AM335x register name	AM335x register bit length	Memory datasheet symbol	Memory Datasheet value	unit	AM335x Setting (Decimal)	Comments
4		tCK		tCK	2.5	ns		speed at which the part will run, not the max freq.
5	SDRAM_TIM_1	REG_T_RP	4	tRP	13.75	ns	5	typically taken from the speed bin tables
6		REG_T_RCD	4	tRCD	13.75	ns	5	typically taken from the speed bin tables
7		REG_T_WR	4	tWR	15	ns	5	
8		REG_T_RAS	5	tRAS	35	ns	13	tRAS should be >= tRCD
9		REG_T_RC	6	tRC	48.75	ns	19	
10		REG_T_RRD	3	tRRD	4	tCK	3	use the value given in CK units
11		REG_T_WTR	3	tWTR	4	tCK	3	use the value given in CK units
12	SDRAM_TIM_2	REG_T_XP	3	tXP	3	tCK	2	use the value given in CK units
13		REG_T_ODT	3	ODTLon	3	tCK	3	typically in terms of CWL. First determine CWL.
14		REG_T_XSNR	9	tXS	270	ns	107	usually tRFC+10
15		REG_T_XSRD	10	tXSDLL	512	tCK	511	usually in terms of tDLLK
16		REG_T_RTP	3	tRTP	4	tCK	3	use the value given in CK units
17		REG_T_CKE	3	tCKE	3	tCK	2	use the value given in CK units
18	SDRAM_TIM_3	REG_T_PDLL_UL	4				5	set to fixed value of 5
19		REG_T_ZQCS	6	tZQCS	64	tCK	63	
20		REG_T_RFC	9	tRFC	260	ns	103	
21				tREFI				
22		REG_T_RAS_MAX	4	tRASmax			15	for DDR3, must be set to 15

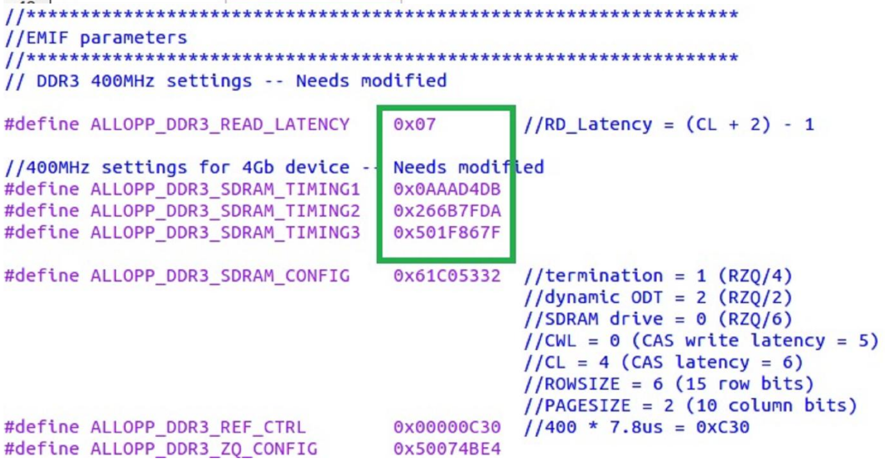
IMPORTANT

The timings above were derived for the **Micron MT41K256M16HA-125:E** which is used on the BeagleBoneBlack. A different memory device will require different timing parameters than used above

2. After ensuring that all of the parameters are entered for the correct memory device grab the values in the **three SDRAM_TIM_n** registers and replace the **ALLOPP_DDR3_SDRAM_TIMINGn** values in the GEL file
   - The values you need to modify are on lines 694-696
   - You will also need to determine the Read Latency by using the formula in the comment on 691. As our CL=6 for this memory at 400MHz, we will **enter in 0x07**



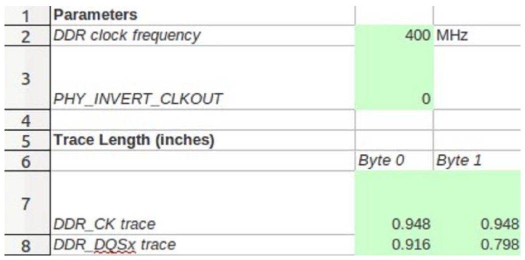
25	SDRAM_TIM_1	
26		reserved[31:29]
27	Bit field values (hex)	0
28	Bit field values (binary)	000
29	Register value (hex) optimized	0AAAD4DB
30		
33		
34	SDRAM_TIM_2	
35		reserved[31]
36	Bit field values (hex)	0
37	Bit field values (binary)	0
38	Register value (hex) optimized	266B7FDA
39		
42		
43	SDRAM_TIM_3	
44		REG_T_PDLL_UL[31:28]
45	Bit field values (hex)	5
46	Bit field values (binary)	0101
47	Register value (hex) optimized	501F867F



```
//*********************************************************************
//EMIF parameters
//*********************************************************************
// DDR3 400MHz settings -- Needs modified

#define ALLOPP_DDR3_READ_LATENCY       0x07        //RD_Latency = (CL + 2) - 1

//400MHz settings for 4Gb device -- Needs modified
#define ALLOPP_DDR3_SDRAM_TIMING1      0x0AAAD4DB
#define ALLOPP_DDR3_SDRAM_TIMING2      0x266B7FDA
#define ALLOPP_DDR3_SDRAM_TIMING3      0x501F867F

#define ALLOPP_DDR3_SDRAM_CONFIG       0x61C05332  //termination = 1 (RZQ/4)
                                                   //dynamic ODT = 2 (RZQ/2)
                                                   //SDRAM drive = 0 (RZQ/6)
                                                   //CWL = 0 (CAS write latency = 5)
                                                   //CL = 4 (CAS latency = 6)
                                                   //ROWSIZE = 6 (15 row bits)
                                                   //PAGESIZE = 2 (10 column bits)
#define ALLOPP_DDR3_REF_CTRL           0x00000C30  //400 * 7.8us = 0xC30
#define ALLOPP_DDR3_ZQ_CONFIG          0x50074BE4
```

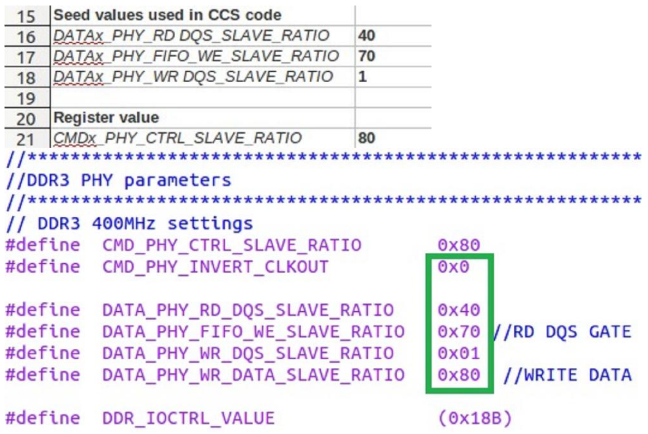
Open the Ratio Seed spreadsheet

1. Specify the speed at which you intend to run the memory (in MHz)
2. Enter the trace lengths for the **DDR_CLK for each byte**
   - If using a single chip use the same length for both. If using two memories with 'T' Routing specify the value for each byte. If using fly-by topology where the trace runs from the first memory to the second then enter the total trace length for both.
3. Enter the DQS trace lengths for each byte **in inches**
   - We defaulted to the positive signal. We used the trace lengths for **DDR_DQS0 and DDR_DQS1** and ignored the negative signals for both bytes
4. Determine whether to enter a '0' or '1' for the PHY_INVERT_CLKOUT
   - If the clock trace is longer than the DQS trace use a '0' – otherwise use a '1'



1	**Parameters**		
2	*DDR clock frequency*	400	MHz
3	*PHY_INVERT_CLKOUT*	0	
4			
5	**Trace Length (inches)**		
6		*Byte 0*	*Byte 1*
7	*DDR_CK trace*	0.948	0.948
8	*DDR_DQSx trace*	0.916	0.798

5. You should see **four values** generated at the bottom plus the **PHY_INVERT_CLKOUT** parameter which you can enter into your GEL file
   - The values you will need to modify are on lines 676-682
   - Although the spreadsheet does not calculate an initial value for this parameter use **0x80 for the WR_DATA_SLAVE_RATIO**. This will be updated by the tuning software.



15	**Seed values used in CCS code**	
16	*DATAx_PHY_RD DQS_SLAVE_RATIO*	**40**
17	*DATAx_PHY_FIFO_WE_SLAVE_RATIO*	**70**
18	*DATAx_PHY_WR DQS_SLAVE_RATIO*	**1**
19		
20	**Register value**	
21	*CMDx_PHY_CTRL_SLAVE_RATIO*	**80**

```
//**************************************************************
//DDR3 PHY parameters
//**************************************************************
// DDR3 400MHz settings
#define  CMD_PHY_CTRL_SLAVE_RATIO         0x80
#define  CMD_PHY_INVERT_CLKOUT            0x0

#define  DATA_PHY_RD_DQS_SLAVE_RATIO      0x40
#define  DATA_PHY_FIFO_WE_SLAVE_RATIO     0x70 //RD DQS GATE
#define  DATA_PHY_WR_DQS_SLAVE_RATIO      0x01
#define  DATA_PHY_WR_DATA_SLAVE_RATIO     0x80  //WRITE DATA

#define  DDR_IOCTRL_VALUE                 (0x18B)
```

Configure the Development Environment

Description

This section will take you through the steps involved in setting up CCS to communicate with the board as well as running the calibration software. Note that some steps may be slightly different depending on specific CCS version and OS.

Prerequisites

- BeagleBone Black
  - We used a rev A5a board
- JTAG Emulator
  - We have tested with a Spectrum Digital XDS560v2 STM USB, BlackHawk XDS560v2 USB, SD XDS510, TI XDS100v2
- Host computer running Code Composer Studio
  - This lab was tested with CCS v5.3, but while other versions may work some edits may be required
  - We tested with the host PC running Windows 7 and Ubuntu 12.04. Other OSes may work, but again some tweaks may be necessary
- Executable CCS .out file
  - This is the software that will determine optimal DDR3 timings

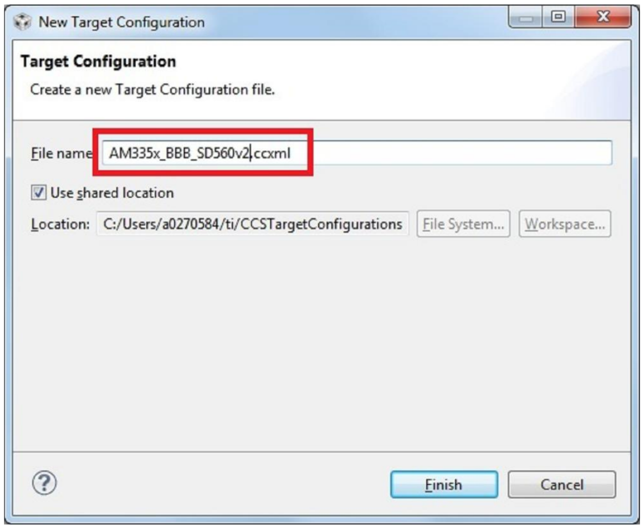
Lab Steps

Configuring the Target Configuration File

IMPORTANT

The BeagleBoneBlack comes with SPL, u-boot and the Linux kernel already burned into its eMMC. Because eMMC is the default boot method on the BBB whenever you provide power it will boot into Linux. We want to prevent this so as to ensure that the board is stuck in the Boot ROM. This has the bonus effect of keeping the device in a 'pure' state. To do this **hold down the S2 button** when applying power. You can let it go after a second or so. If you are connected to the serial pins you can see the serial output spitting out 'C' characters. This is the ROM telling us that it is waiting for a valid SPL image from an SD card. **If you see any of the User LEDs (opposite the RJ45 connector from the power plug) lit up blue this means that Linux has booted.**

1. Select **File -> New -> Target Configuration File**
2. In the **New Target Configuration** dialog box give the configuration a name. For this lab we will use AM335x_BBB_SD560v2
   - It is strongly encouraged to create a meaningful name (e.g. Processor_BoardName_EmulatorType.ccxml)


New Target Configuration
Target Configuration
Create a new Target Configuration file.
File name AM335x_BBB_SD560v2.ccxml
Use shared location
Location: C:/Users/a0270584/ti/CCSTargetConfigurations
File System...
Workspace...
Finish
Cancel

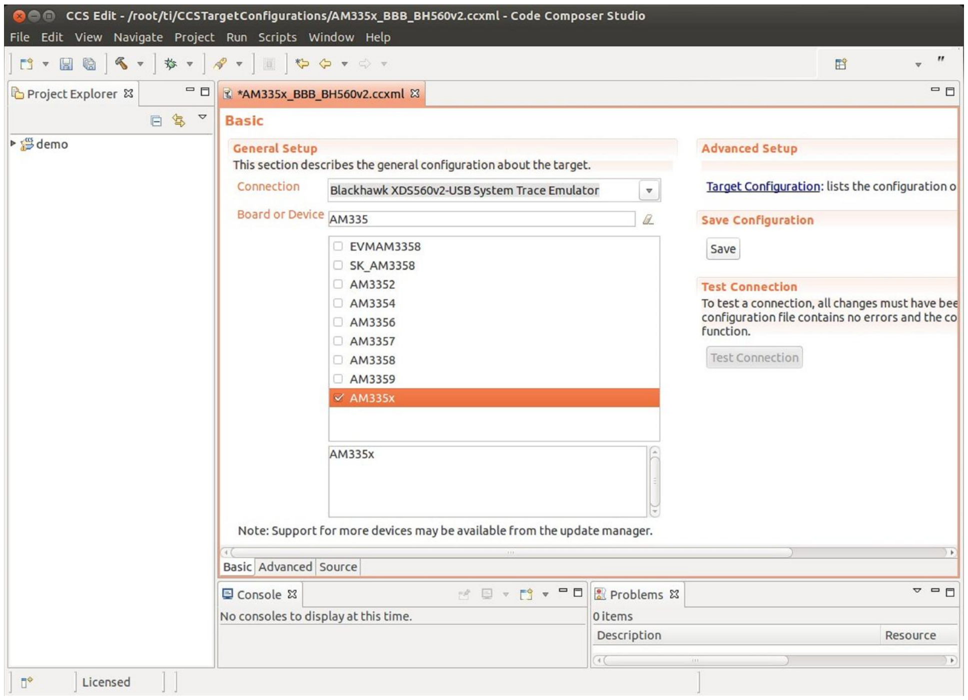
3. Click **Use shared location**, then Click **Finish**
4. The *AM335x_BBB_SD560v2* configuration you created will now be opened for editing. Perform the following steps to finish the target configuration.
   - In the **Connection** drop-down box select **your emulator type.**

     The pictures below use a Spectrum Digital XDS560V2 STM USB Emulator, but you will need to select the emulator you are using.

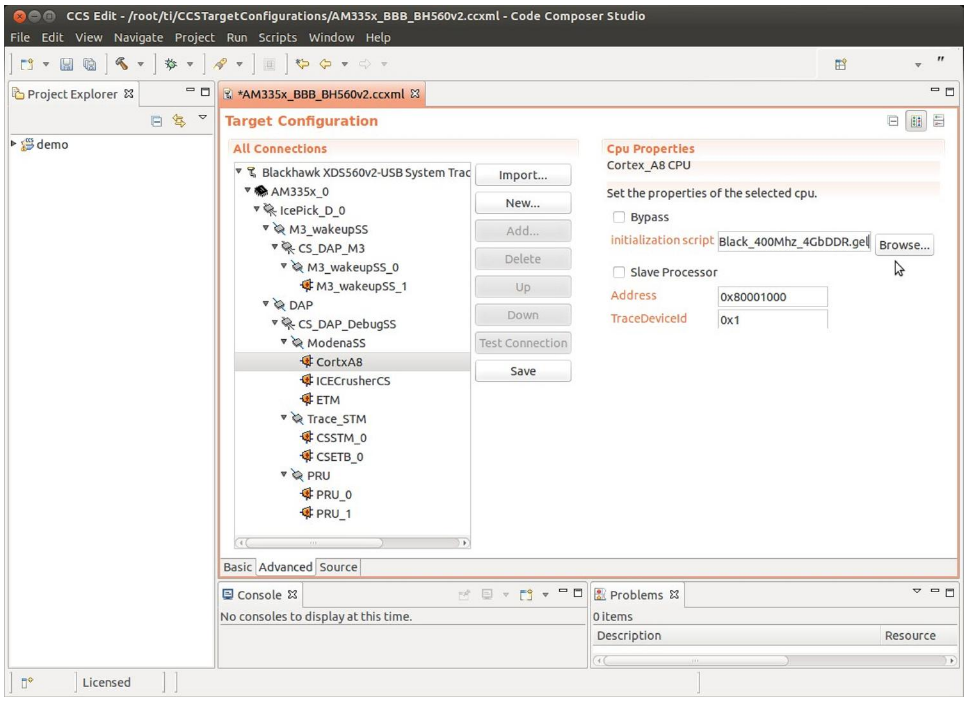
   - **NOTE**

     If you are using a different emulator please select that one instead. You will only see emulators for which you have installed the drivers

   - In the **Board or Device** selection list check the **AM335x** device
   - You screen should look like

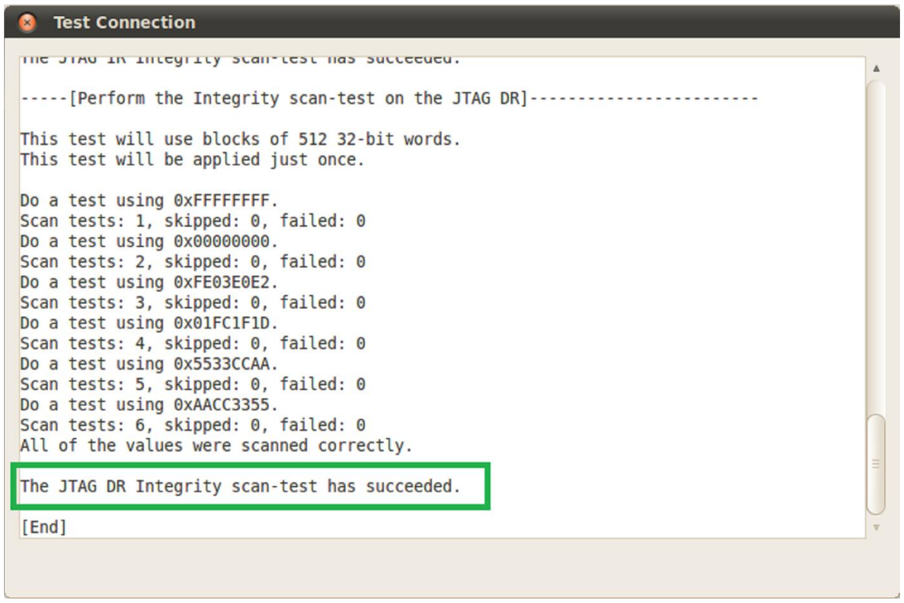

CCS Edit - /root/ti/CCSTargetConfigurations/AM335x_BBB_BH560v2.ccxml - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
demo
*AM335x_BBB_BH560v2.ccxml
Basic
General Setup
This section describes the general configuration about the target.
Connection Blackhawk XDS560v2-USB System Trace Emulator
Board or Device AM335
EVMAM3358
SK_AM3358
AM3352
AM3354
AM3356
AM3357
AM3358
AM3359
AM335x
AM335x
Note: Support for more devices may be available from the update manager.
Advanced Setup
Target Configuration: lists the configuration o
Save Configuration
Save
Test Connection
To test a connection, all changes must have bee configuration file contains no errors and the co function.
Test Connection
Basic Advanced Source
Console
No consoles to display at this time.
Problems
0 items
Description
Resource
Licensed

5. Select the Advanced tab, highlight the CortxA8 chip and point the initialization script field to the GEL file you modified in the previous section



- Click the **Save** button at the right of the screen

6. You can now test your target connection by pressing the **Test Connection** button at the right of the screen.
   - You should see a dialog box open. Scroll to the bottom of the output and look for the **The JTAG DR Integrity scan-test has succeeded**



```
The JTAG IR Integrity scan-test has succeeded.

-----[Perform the Integrity scan-test on the JTAG DR]------------------------

This test will use blocks of 512 32-bit words.
This test will be applied just once.

Do a test using 0xFFFFFFFF.
Scan tests: 1, skipped: 0, failed: 0
Do a test using 0x00000000.
Scan tests: 2, skipped: 0, failed: 0
Do a test using 0xFE03E0E2.
Scan tests: 3, skipped: 0, failed: 0
Do a test using 0x01FC1F1D.
Scan tests: 4, skipped: 0, failed: 0
Do a test using 0x5533CCAA.
Scan tests: 5, skipped: 0, failed: 0
Do a test using 0xAACC3355.
Scan tests: 6, skipped: 0, failed: 0
All of the values were scanned correctly.

The JTAG DR Integrity scan-test has succeeded.

[End]
```

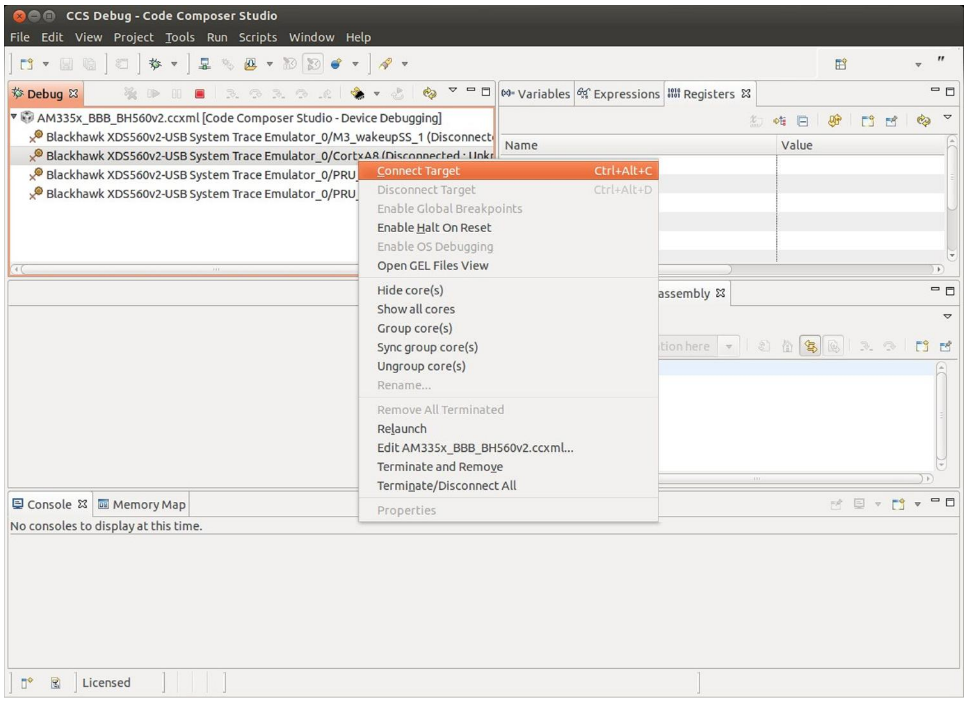
> **NOTE**
>
> There are other tests run during the connection test and the status of these tests is also displayed in the dialog box

> **NOTE**
>
> The JTAG scan test may take a few seconds to complete on slower laptops.

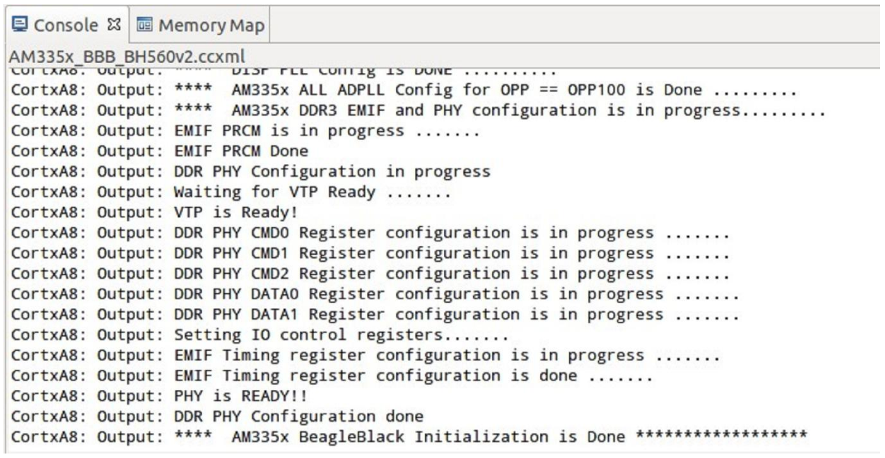
7. Close the *Test Connection* dialog box
   - The target connection is now created

Tuning the DDR3 Timings

1. Open the Target Configurations window (**View->Target Configurations**)
2. Locate your new Target Configuration file, right click it and select **Launch Selected Configuration**
   - When the Debug Perspective finishes loading you will notice several cores in the Debug window
   - While these are all technically debuggable, the only one we care about is the Cortex-A8
3. Connect to the CortxA8 by right clicking it and selecting **Connect to Target**

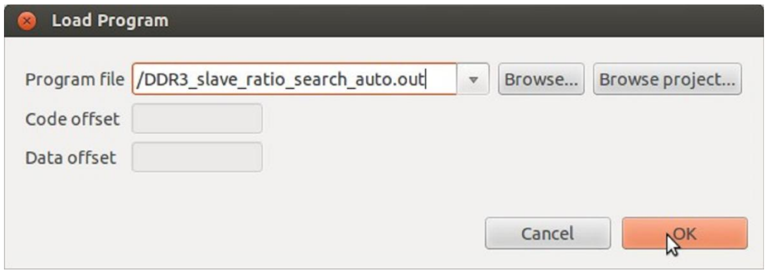

CCS Debug - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Variables Expressions Registers
Name Value
AM335x_BBB_BH560v2.ccxml [Code Composer Studio - Device Debugging]
Blackhawk XDS560v2-USB System Trace Emulator_0/M3_wakeupSS_1 (Disconnected
Blackhawk XDS560v2-USB System Trace Emulator_0/CortxA8 (Disconnected : Unkn
Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_
Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_
Connect Target Ctrl+Alt+C
Disconnect Target Ctrl+Alt+D
Enable Global Breakpoints
Enable Halt On Reset
Enable OS Debugging
Open GEL Files View
Hide core(s)
Show all cores
Group core(s)
Sync group core(s)
Ungroup core(s)
Rename...
Remove All Terminated
Relaunch
Edit AM335x_BBB_BH560v2.ccxml...
Terminate and Remove
Terminate/Disconnect All
Properties
assembly
Console Memory Map
No consoles to display at this time.
Licensed

- As it is connecting you will see a slew of messages printed out from the GEL file



```
CortxA8: Output: ****  DISP PLL Config is DONE ..........
CortxA8: Output: ****  AM335x ALL ADPLL Config for OPP == OPP100 is Done .........
CortxA8: Output: ****  AM335x DDR3 EMIF and PHY configuration is in progress.........
CortxA8: Output: EMIF PRCM is in progress .......
CortxA8: Output: EMIF PRCM Done
CortxA8: Output: DDR PHY Configuration in progress
CortxA8: Output: Waiting for VTP Ready .......
CortxA8: Output: VTP is Ready!
CortxA8: Output: DDR PHY CMD0 Register configuration is in progress .......
CortxA8: Output: DDR PHY CMD1 Register configuration is in progress .......
CortxA8: Output: DDR PHY CMD2 Register configuration is in progress .......
CortxA8: Output: DDR PHY DATA0 Register configuration is in progress .......
CortxA8: Output: DDR PHY DATA1 Register configuration is in progress .......
CortxA8: Output: Setting IO control registers.......
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 BeagleBlack Initialization is Done ******************
```

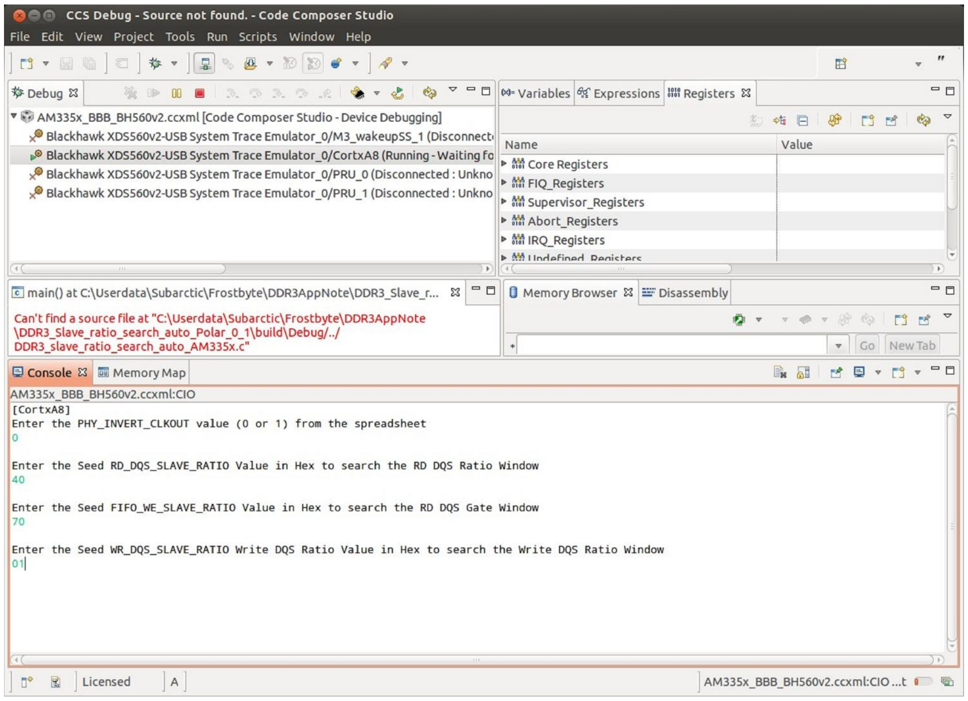
4. Load the DDR3_slave_ratio_search_auto.out file (**Run->Load Program**)



Program file: /DDR3_slave_ratio_search_auto.out
Code offset:
Data offset:

- When the file finishes loading you will see a window stating that CCS cannot find the source file. This is normal because we do not have the project loaded into the workspace, and therefore no source is available to the debugger

5. Run the code by pressing the **green arrow button**
6. You will see some prompts in the console. At this point you will need to **enter the hex values you calculated in the Ratio Seed spreadsheet** (e.g. for 0x40 enter in 40)


CCS Debug - Source not found. - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Variables Expressions Registers
Name Value
Core Registers
FIQ_Registers
Supervisor_Registers
Abort_Registers
IRQ_Registers
Memory Browser Disassembly
Go New Tab
Console Memory Map
Licensed A
AM335x_BBB_BH560v2.ccxml:CIO ...t

```
AM335x_BBB_BH560v2.ccxml [Code Composer Studio - Device Debugging]
  Blackhawk XDS560v2-USB System Trace Emulator_0/M3_wakeupSS_1 (Disconnect
  Blackhawk XDS560v2-USB System Trace Emulator_0/CortxA8 (Running - Waiting fo
  Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_0 (Disconnected : Unkno
  Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_1 (Disconnected : Unkno
```

```
main() at C:\Userdata\Subarctic\Frostbyte\DDR3AppNote\DDR3_Slave_r...
Can't find a source file at "C:\Userdata\Subarctic\Frostbyte\DDR3AppNote
\DDR3_Slave_ratio_search_auto_Polar_0_1\build\Debug/../
DDR3_slave_ratio_search_auto_AM335x.c"
```

```
AM335x_BBB_BH560v2.ccxml:CIO
[CortxA8]
Enter the PHY_INVERT_CLKOUT value (0 or 1) from the spreadsheet
0

Enter the Seed RD_DQS_SLAVE_RATIO Value in Hex to search the RD DQS Ratio Window
40

Enter the Seed FIFO_WE_SLAVE_RATIO Value in Hex to search the RD DQS Gate Window
70

Enter the Seed WR_DQS_SLAVE_RATIO Write DQS Ratio Value in Hex to search the Write DQS Ratio Window
01
```

NOTE

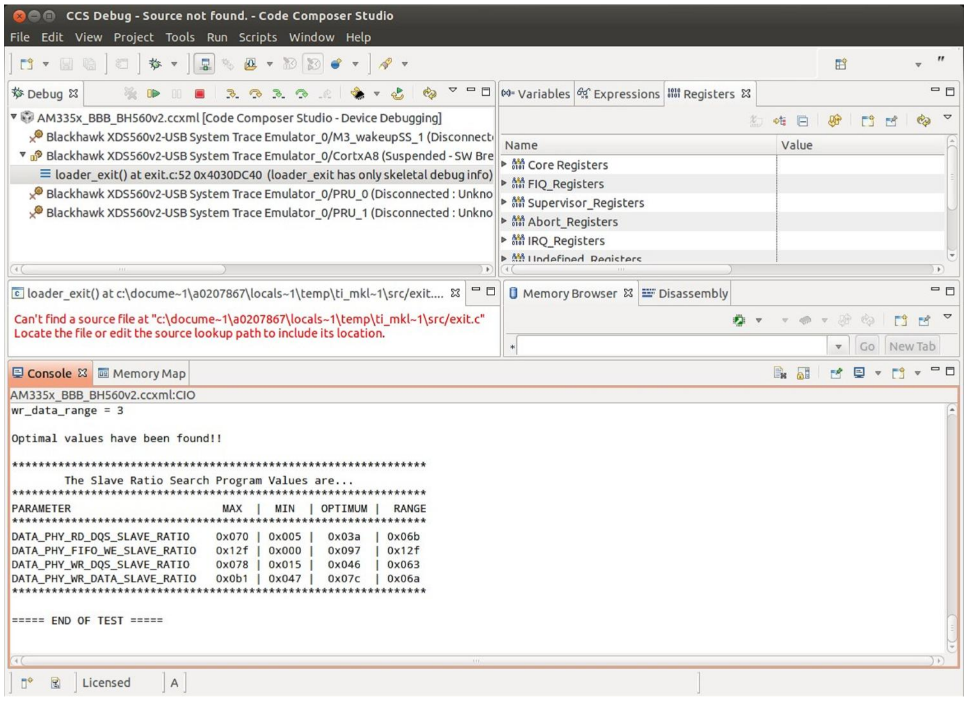
You may need to click in the console window area before it will allow you to input these numbers

7. The software will begin calibrating
   - While it is tuning itself you will see several iterations of output as it hones in on optimal values

IMPORTANT

If the console outputs all 0x0 values for the minimum, maximum, optimum and range columns then it did not run correctly. This is because one or more of the parameters in the GEL file were not updated or you did not load the correct values into the console when you ran the .out program.

8. Once this finishes you have your final DDR3 configuration values in the **OPTIMUM column**

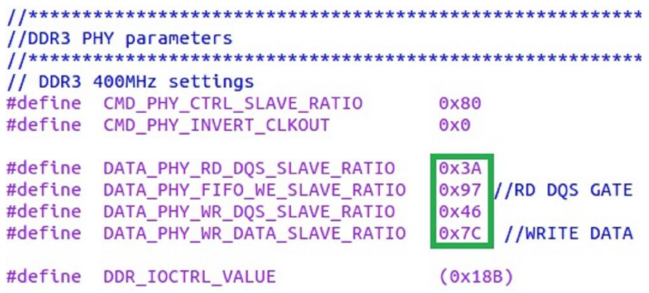

CCS Debug - Source not found. - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Variables Expressions Registers
Name Value
Core Registers
FIQ_Registers
Supervisor_Registers
Abort_Registers
IRQ_Registers
Memory Browser Disassembly
Go New Tab
Console Memory Map
Licensed A

```
AM335x_BBB_BH560v2.ccxml [Code Composer Studio - Device Debugging]
  Blackhawk XDS560v2-USB System Trace Emulator_0/M3_wakeupSS_1 (Disconnect
  Blackhawk XDS560v2-USB System Trace Emulator_0/CortxA8 (Suspended - SW Bre
    loader_exit() at exit.c:52 0x4030DC40 (loader_exit has only skeletal debug info)
  Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_0 (Disconnected : Unkno
  Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_1 (Disconnected : Unkno
```

```
loader_exit() at c:\docume~1\a0207867\locals~1\temp\ti_mkl~1\src/exit....
Can't find a source file at "c:\docume~1\a0207867\locals~1\temp\ti_mkl~1\src/exit.c"
Locate the file or edit the source lookup path to include its location.
```

```
AM335x_BBB_BH560v2.ccxml:CIO
wr_data_range = 3

Optimal values have been found!!

**************************************************************
        The Slave Ratio Search Program Values are...
**************************************************************
PARAMETER                       MAX  |  MIN  | OPTIMUM |  RANGE
**************************************************************
DATA_PHY_RD_DQS_SLAVE_RATIO    0x070 | 0x005 |  0x03a  | 0x06b
DATA_PHY_FIFO_WE_SLAVE_RATIO   0x12f | 0x000 |  0x097  | 0x12f
DATA_PHY_WR_DQS_SLAVE_RATIO    0x078 | 0x015 |  0x046  | 0x063
DATA_PHY_WR_DATA_SLAVE_RATIO   0x0b1 | 0x047 |  0x07c  | 0x06a
**************************************************************

===== END OF TEST =====
```

9. **Copy these OPTIMUM values** into the corresponding locations inside the GEL file and then **terminate the CCS session**



```
//*********************************************************
//DDR3 PHY parameters
//*********************************************************
// DDR3 400MHz settings
#define  CMD_PHY_CTRL_SLAVE_RATIO       0x80
#define  CMD_PHY_INVERT_CLKOUT          0x0

#define  DATA_PHY_RD_DQS_SLAVE_RATIO    0x3A
#define  DATA_PHY_FIFO_WE_SLAVE_RATIO   0x97 //RD DQS GATE
#define  DATA_PHY_WR_DQS_SLAVE_RATIO    0x46
#define  DATA_PHY_WR_DATA_SLAVE_RATIO   0x7C  //WRITE DATA

#define  DDR_IOCTRL_VALUE               (0x18B)
```

NOTE

Do not be alarmed if the values you receive between tests or between boards from the same spin are slightly different. This is perfectly acceptable unless there are some large differences (e.g., >0x10) between them.

Test the DDR3 Configuration

Description

This section will cover how to quickly test out the new DDR3 configuration.

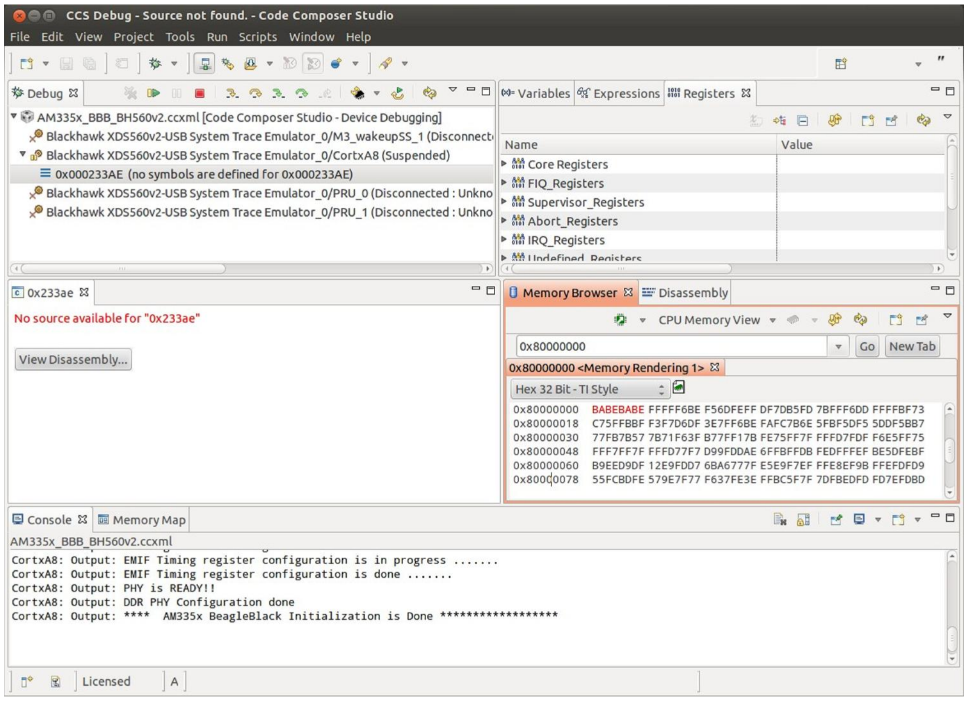
Prerequisites

- BeagleBone Black
- JTAG Emulator
  - We tested with a BlackHawk XDS560v2, but an XDS510, 100v2, or other similar emulator should work fine
- Host computer running Code Composer Studio
  - This lab was tested with CCS v5.3, but while other versions may work some edits may be required
  - We tested with the host PC running Windows 7 and Ubuntu 12.04. Other OSes may work, but again some tweaks may be necessary

Lab Steps

It is strongly recommended to power cycle the board. This is to ensure that the board is in a default, known-good configuration so that the only modifications to any registers are what the newly optimized GEL file changes.

1. Launch the **Target Configuration** you created in the previous section and **connect** to the Cortex-A8 core
2. Open the memory browser (**View->Memory Browser**)
3. Point it to the **DDR3 memory**. This range starts at 0x80000000 and ends at 0xBFFFFFFFF
4. Try to *modify the memory'* by double-clicking on a value and changing it to something else
   - If the memory is configured correctly you should see your value 'stick' every time



```
AM335x_BBB_BH560v2.ccxml [Code Composer Studio - Device Debugging]
  Blackhawk XDS560v2-USB System Trace Emulator_0/M3_wakeupSS_1 (Disconnect
  Blackhawk XDS560v2-USB System Trace Emulator_0/CortxA8 (Suspended)
    0x000233AE  (no symbols are defined for 0x000233AE)
  Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_0 (Disconnected : Unkno
  Blackhawk XDS560v2-USB System Trace Emulator_0/PRU_1 (Disconnected : Unkno
```

No source available for "0x233ae"

```
0x80000000
0x80000000   BABEBABE FFFFF6BE F56DFEFF DF7DB5FD 7BFFF6DD FFFFBF73
0x80000018   C75FFBBF F3F7D6DF 3E7FF6BE FAFC7B6E 5FBF5DF5 5DDF5BB7
0x80000030   77FB7B57 7B71F63F B77FF17B FE75FF7F FFFD7FDF F6E5FF75
0x80000048   FFF7FF7F FFFD77F7 D99FDDAE 6FFBFFDB FEDFFFEF BE5DFEBF
0x80000060   B9EED9DF 12E9FDD7 6BA6777F E5E9F7EF FFE8EF9B FFEFDFD9
0x80000078   55FCBDFE 579E7F77 F637FE3E FFBC5F7F 7DFBEDFD FD7EFDBD
```

```
AM335x_BBB_BH560v2.ccxml
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 BeagleBlack Initialization is Done ******************
```

NOTE

This is a simple test, and usually only helps to check for catastrophic problems such as incorrect EMIF register values

5. For a more extensive test run the EDMA test from the GEL script (**Scripts->AM335x DDR Tests->EDMA**)
   - This will take some time, but it is a much more thorough test

Updating SPL/U-boot

Description

This section discusses how to modify the SPL code within the AM335x SDK to contain the new DDR3 timings.

Prerequisites

- Linux host machine
  - This lab was built on an Ubuntu 12.04 64-bit system
- The sitara-board-port-uboot git tree cloned into a directory
  - In the TI hands on lab this has been cloned for you in the */home/sitara/ti-sdk-am335x-evm-06.00.00.00/board-support/board-port directory*.
  - If you are cloning these sources yourself you can do so using the following command (Assuming you have git properly configured to work with your network proxy if you use one).
    - git clone git://gitorious.org/sitara-board-port/sitara-board-port-uboot.git

Lab Steps

1. Open a terminal and navigate to the root of the u-boot source. In the hands-on training this is /home/sitara/ti-sdk-am335x-evm-06.00.00.00/board-support/board-port-labs/sitara-board-port-uboot/
   - To ensure that you are working with the correct tag type

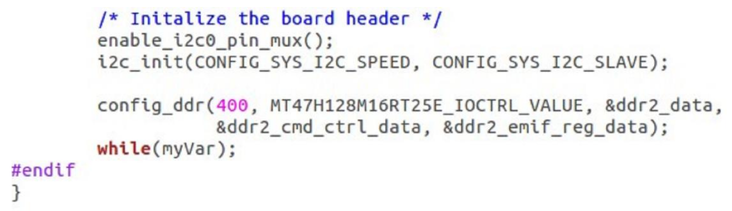
     git checkout 06.00.00.00-template

     > **NOTE**
     >
     > You can type **git tag** to see the other available tags

2. Open the **board.c** file found in board/ti/am335x/ with the editor of your choice
3. Locate the **s_init** function
4. Add a volatile variable at the start of the function, e.g. **volatile int myVar = 1;**

   > **NOTE**
   >
   > The volatile keyword is to prevent the optimizer from removing that variable.

5. Modify the **first parameter** in the config_ddr function call from **266 to 400**
   - This tells SPL that the memory is operating at 400MHz instead of the default of 266MHz



```
        /* Initalize the board header */
        enable_i2c0_pin_mux();
        i2c_init(CONFIG_SYS_I2C_SPEED, CONFIG_SYS_I2C_SLAVE);

        config_ddr(400, MT47H128M16RT25E_IOCTRL_VALUE, &ddr2_data,
                   &ddr2_cmd_ctrl_data, &ddr2_emif_reg_data);
        while(myVar);
#endif
}
```

6. Immediately following the config_ddr() function call add in a **while(myVar);** infinite loop

   > **IMPORTANT**
   >
   > **This is critical** as the u-boot source in this tree is designed for the BeagleBone, and not the BeagleBoneBlack. There were some hardware differences between the two specifically with different voltage levels. Fortunately the initialization of these higher voltage levels takes place after the DDR configuration, so that when we hit that infinite loop we will be safe.

7. **Save and close** the board.c file
8. Open the **ddr_defs.h** header file found in arch/arm/include/asm/arch-am33xx/
9. You will add the timings you determined from the spreadsheets in the first section into this file
   - We are going to create a new set of parameters corresponding to the actual memory device on the BBBlack board (MT41K256M16HA125E_EMIF_READ_LATENCY, etc.)
   - The board.c file actually uses a number of these header definitions to gain the actual EMIF register values.
   - **Copy these parameters** into the ddr_defs.h file

```
/* Micron MT41K256M16HA-125:E - BBBlack */
#define MT41K256M16HA125E_EMIF_READ_LATENCY 0x100007
#define MT41K256M16HA125E_EMIF_TIM1 0x0AAAD4DB
#define MT41K256M16HA125E_EMIF_TIM2 0x266B7FDA
#define MT41K256M16HA125E_EMIF_TIM3 0x501F867F
#define MT41K256M16HA125E_EMIF_SDCFG 0x61C05332
#define MT41K256M16HA125E_EMIF_SDREF 0x00000C30
#define MT41K256M16HA125E_ZQ_CFG 0x50074BE4
#define MT41K256M16HA125E_DLL_LOCK_DIFF 0x1
#define MT41K256M16HA125E_RATIO 0x80
#define MT41K256M16HA125E_INVERT_CLKOUT 0x00
#define MT41K256M16HA125E_RD_DQS 0x3A
```

```
#define MT41K256M16HA125E_WR_DQS 0x45
#define MT41K256M16HA125E_PHY_WR_DATA 0x7C
#define MT41K256M16HA125E_PHY_FIFO_WE 0x96
#define MT41K256M16HA125E_IOCTRL_VALUE 0x18B
```

10. **Save and close** the ddr_defs.h file

    > **NOTE**
    >
    > Because you created the new parameter set you will need to create a few new structures in the board.c to correspond to these new memory parameters.

11. Open the **board/ti/am335x/board.c** file again and add these structures above the am33xx_spl_board_init function around line '306':

```
static const struct ddr_data ddr3_bbb_data = {

        .datardsratio0 = MT41K256M16HA125E_RD_DQS,
        .datawdsratio0 = MT41K256M16HA125E_WR_DQS,
        .datafwsratio0 = MT41K256M16HA125E_PHY_FIFO_WE,
        .datawrsratio0 = MT41K256M16HA125E_PHY_WR_DATA,
        .datadldiff0 = PHY_DLL_LOCK_DIFF,

};
static const struct cmd_control ddr3_bbb_cmd_ctrl_data = {

        .cmd0csratio = MT41K256M16HA125E_RATIO,
        .cmd0dldiff = MT41K256M16HA125E_DLL_LOCK_DIFF,
        .cmd0iclkout = MT41K256M16HA125E_INVERT_CLKOUT,
        .cmd1csratio = MT41K256M16HA125E_RATIO,
        .cmd1dldiff = MT41K256M16HA125E_DLL_LOCK_DIFF,
        .cmd1iclkout = MT41K256M16HA125E_INVERT_CLKOUT,
        .cmd2csratio = MT41K256M16HA125E_RATIO,
        .cmd2dldiff = MT41K256M16HA125E_DLL_LOCK_DIFF,
        .cmd2iclkout = MT41K256M16HA125E_INVERT_CLKOUT,

};
static struct emif_regs ddr3_bbb_emif_reg_data = {

        .sdram_config = MT41K256M16HA125E_EMIF_SDCFG,
        .ref_ctrl = MT41K256M16HA125E_EMIF_SDREF,
        .sdram_tim1 = MT41K256M16HA125E_EMIF_TIM1,
        .sdram_tim2 = MT41K256M16HA125E_EMIF_TIM2,
        .sdram_tim3 = MT41K256M16HA125E_EMIF_TIM3,
        .zq_config = MT41K256M16HA125E_ZQ_CFG,
        .emif_ddr_phy_ctlr_1 = MT41K256M16HA125E_EMIF_READ_LATENCY | PHY_EN_DYN_PWRDN,

};
```

12. Then go **update the config_ddr** function call to look like

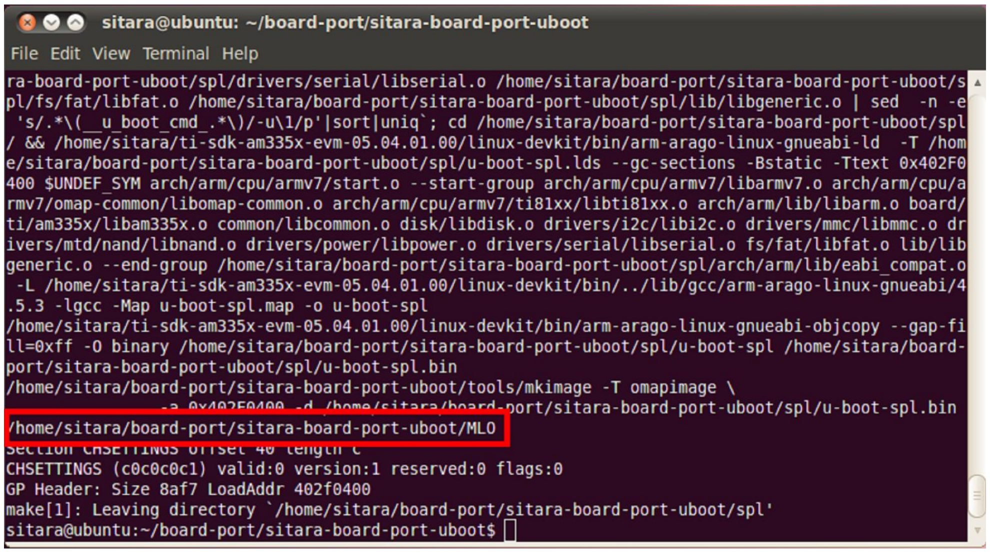
```
config_ddr(400, MT41K256M16HA125E_IOCTRL_VALUE, &ddr3_bbb_data, &ddr3_bbb_cmd_ctrl_data, &ddr3_bbb_emif_reg_data);
```

    > **NOTE**
    >
    > This updates the config_ddr function call to use our new timing parameters for the memory on the BeagleBoneBlack

13. **Save and close** the board.c file
14. After making these changes you will need to rebuild u-boot as well as the board configuration. We can do this in one step by building the *am335x_evm* configuration

```
make ARCH=arm CROSS_COMPILE=/home/sitara/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/i686-arago-linux/usr/bin/arm-linux-gnueabihf- am335x_evm
```

    - When SPL/u-boot are finished building you should see output like


sitara@ubuntu: ~/board-port/sitara-board-port-uboot

```
ra-board-port-uboot/spl/drivers/serial/libserial.o /home/sitara/board-port/sitara-board-port-uboot/s
pl/fs/fat/libfat.o /home/sitara/board-port/sitara-board-port-uboot/spl/lib/libgeneric.o | sed  -n -e
 's/.*\(__u_boot_cmd_.*\)/-u\1/p'|sort|uniq`; cd /home/sitara/board-port/sitara-board-port-uboot/spl
/ && /home/sitara/ti-sdk-am335x-evm-05.04.01.00/linux-devkit/bin/arm-arago-linux-gnueabi-ld  -T /hom
e/sitara/board-port/sitara-board-port-uboot/spl/u-boot-spl.lds --gc-sections -Bstatic -Ttext 0x402F0
400 $UNDEF_SYM arch/arm/cpu/armv7/start.o --start-group arch/arm/cpu/armv7/libarmv7.o arch/arm/cpu/a
rmv7/omap-common/libomap-common.o arch/arm/cpu/armv7/ti81xx/libti81xx.o arch/arm/lib/libarm.o board/
ti/am335x/libam335x.o common/libcommon.o disk/libdisk.o drivers/i2c/libi2c.o drivers/mmc/libmmc.o dr
ivers/mtd/nand/libnand.o drivers/power/libpower.o drivers/serial/libserial.o fs/fat/libfat.o lib/lib
generic.o --end-group /home/sitara/board-port/sitara-board-port-uboot/spl/arch/arm/lib/eabi_compat.o
 -L /home/sitara/ti-sdk-am335x-evm-05.04.01.00/linux-devkit/bin/../lib/gcc/arm-arago-linux-gnueabi/4
.5.3 -lgcc -Map u-boot-spl.map -o u-boot-spl
/home/sitara/ti-sdk-am335x-evm-05.04.01.00/linux-devkit/bin/arm-arago-linux-gnueabi-objcopy --gap-fi
ll=0xff -O binary /home/sitara/board-port/sitara-board-port-uboot/spl/u-boot-spl /home/sitara/board-
port/sitara-board-port-uboot/spl/u-boot-spl.bin
/home/sitara/board-port/sitara-board-port-uboot/tools/mkimage -T omapimage \
                -a 0x402F0400 -d /home/sitara/board-port/sitara-board-port-uboot/spl/u-boot-spl.bin
/home/sitara/board-port/sitara-board-port-uboot/MLO
Section CHSETTINGS offset 40 length c
CHSETTINGS (c0c0c0c1) valid:0 version:1 reserved:0 flags:0
GP Header: Size 8af7 LoadAddr 402f0400
make[1]: Leaving directory `/home/sitara/board-port/sitara-board-port-uboot/spl'
sitara@ubuntu:~/board-port/sitara-board-port-uboot$
```

Quickly Test New SPL With CCS

Description

This section will instruct you how to quickly test your changes to SPL without needing an SD card by loading the binary into memory directly through the CCS debugger.

Prerequisites

- u-boot-spl file
  - This was built in the previous section and is located in the spl sub-directory
- Host computer with CCS
  - We used version v5.3, but others may or may not work the same

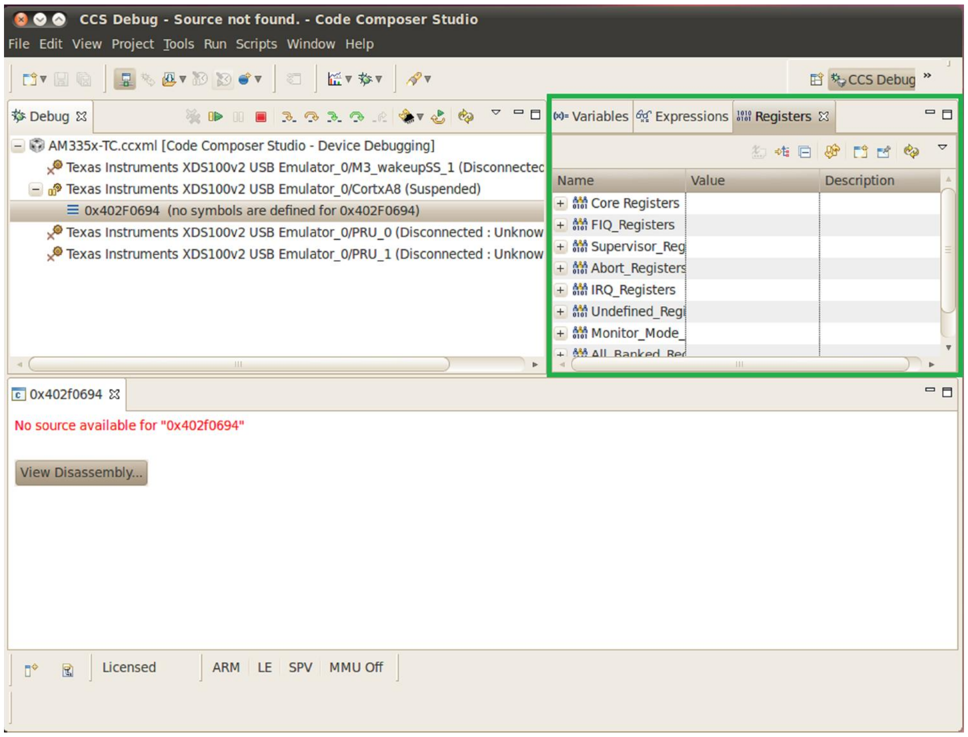
Lab Steps

Before applying power to the BeagleBone Black board you will need to hold the User button (S2) down. This is to force the AM335x to attempt to boot from the SD card instead of the on-board eMMC. These boards have a working spl, u-boot and Linux kernel already, so if we allow them to boot a number of different registers will have been modified before we even connect through JTAG.

1. Open CCS and edit your Target Configuration File
   - You will need to remove the GEL file in the Advanced tab so that the processor is in an unaltered condition prior to running the SPL code
   - Save and close the file
2. Launch the Target Configuration you created in the previous section and connect to the Cortex-A8 core
3. Now that you have connected you can set the processor in ARM Mode with CCS

   > **IMPORTANT**
   >
   > The ROM code runs in Thumb2 Mode and before the SPL code can be run the processor must be set to ARM mode. This would normally be done by the ROM code, but since we are bypassing the ROM load of the SPL we must do it ourselves

   - Select the **Registers** tab


CCS Debug - Source not found. - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
CCS Debug
Debug
Variables
Expressions
Registers
View Disassembly...
Licensed
ARM
LE
SPV
MMU Off

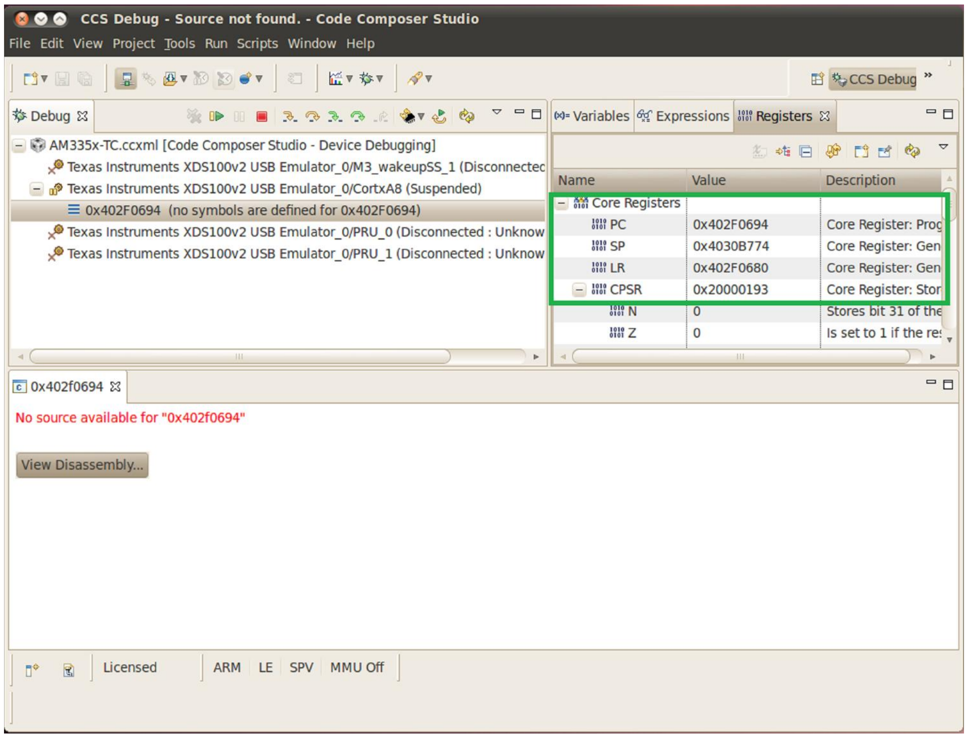
AM335x-TC.ccxml [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/M3_wakeupSS_1 (Disconnectec
Texas Instruments XDS100v2 USB Emulator_0/CortxA8 (Suspended)
0x402F0694 (no symbols are defined for 0x402F0694)
Texas Instruments XDS100v2 USB Emulator_0/PRU_0 (Disconnected : Unknow
Texas Instruments XDS100v2 USB Emulator_0/PRU_1 (Disconnected : Unknow

Name	Value	Description
Core Registers		
FIQ_Registers		
Supervisor_Reg		
Abort_Registers		
IRQ_Registers		
Undefined_Regi		
Monitor_Mode_		
All_Banked_Reg		

0x402f0694

No source available for "0x402f0694"

- Expand the **Core Registers**
- Inside of **Core Registers** expand the **CPSR** list


CCS Debug - Source not found. - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
CCS Debug
Debug
Variables
Expressions
Registers
View Disassembly...
Licensed
ARM
LE
SPV
MMU Off

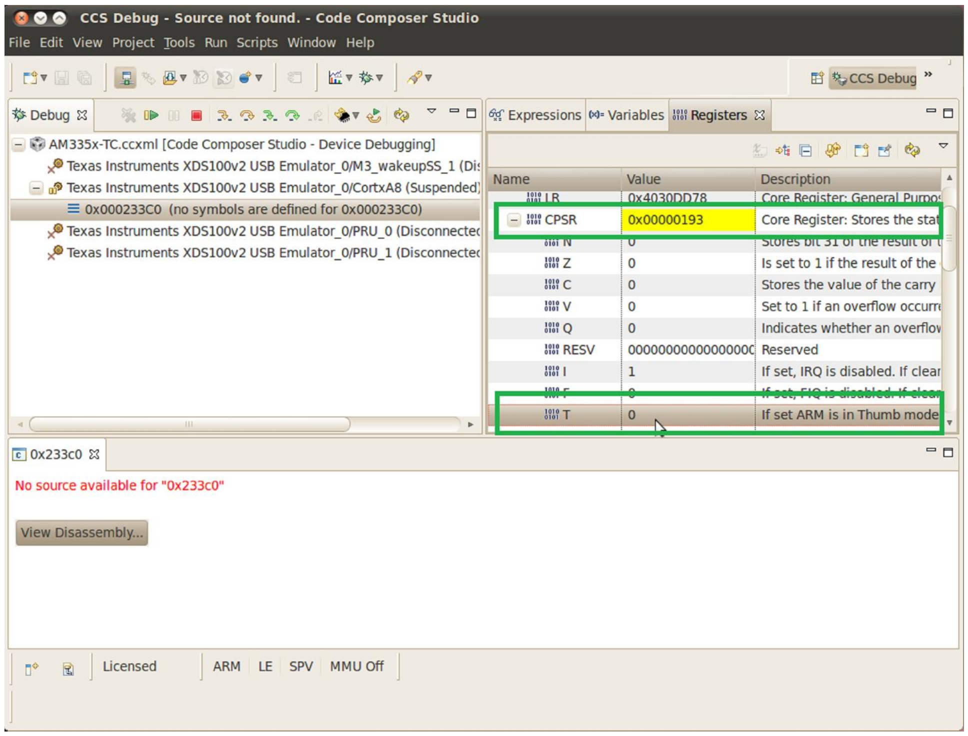
AM335x-TC.ccxml [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/M3_wakeupSS_1 (Disconnectec
Texas Instruments XDS100v2 USB Emulator_0/CortxA8 (Suspended)
0x402F0694 (no symbols are defined for 0x402F0694)
Texas Instruments XDS100v2 USB Emulator_0/PRU_0 (Disconnected : Unknow
Texas Instruments XDS100v2 USB Emulator_0/PRU_1 (Disconnected : Unknow

Name	Value	Description
Core Registers		
PC	0x402F0694	Core Register: Prog
SP	0x4030B774	Core Register: Gen
LR	0x402F0680	Core Register: Gen
CPSR	0x20000193	Core Register: Stor
N	0	Stores bit 31 of the
Z	0	Is set to 1 if the res

0x402f0694

No source available for "0x402f0694"

- Scroll down in the CPSR register list and change the **T** register from **1** to **0**. This is done by
  - Clicking the the **Value** box for the T register where the 1 located
  - Pressing **delete** to remove the value
  - Entering **0**
  - Pressing **Enter**
- You should see the CPSR value change to reflect the new value


CCS Debug - Source not found. - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
CCS Debug
Debug
Expressions
Variables
Registers
0x233c0
View Disassembly...
Licensed
ARM
LE
SPV
MMU Off

AM335x-TC.ccxml [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/M3_wakeupSS_1 (Di
Texas Instruments XDS100v2 USB Emulator_0/CortxA8 (Suspended)
0x000233C0 (no symbols are defined for 0x000233C0)
Texas Instruments XDS100v2 USB Emulator_0/PRU_0 (Disconnecte
Texas Instruments XDS100v2 USB Emulator_0/PRU_1 (Disconnecte

Name	Value	Description
LR	0x4030DD78	Core Register: General Purpo
CPSR	0x00000193	Core Register: Stores the stat
N	0	Stores bit 31 of the result of t
Z	0	Is set to 1 if the result of the
C	0	Stores the value of the carry
V	0	Set to 1 if an overflow occurr
Q	0	Indicates whether an overflow
RESV	0000000000000000	Reserved
I	1	If set, IRQ is disabled. If clear
F	0	If set, FIQ is disabled. If clear
T	0	If set ARM is in Thumb mode

No source available for "0x233c0"

IMPORTANT

You need to do this each time the board is powered completely off or reset. If you use the JTAG to do a SW reset this value will be preserved as well. Likewise, if you attach to a board that was already running a booted system this value will be configured properly but you should always go ahead and check it just to be safe

4. You can now load the SPL binary into the internal chip memory. This is done using the following steps
   - Click **Tools -> Load Memory**

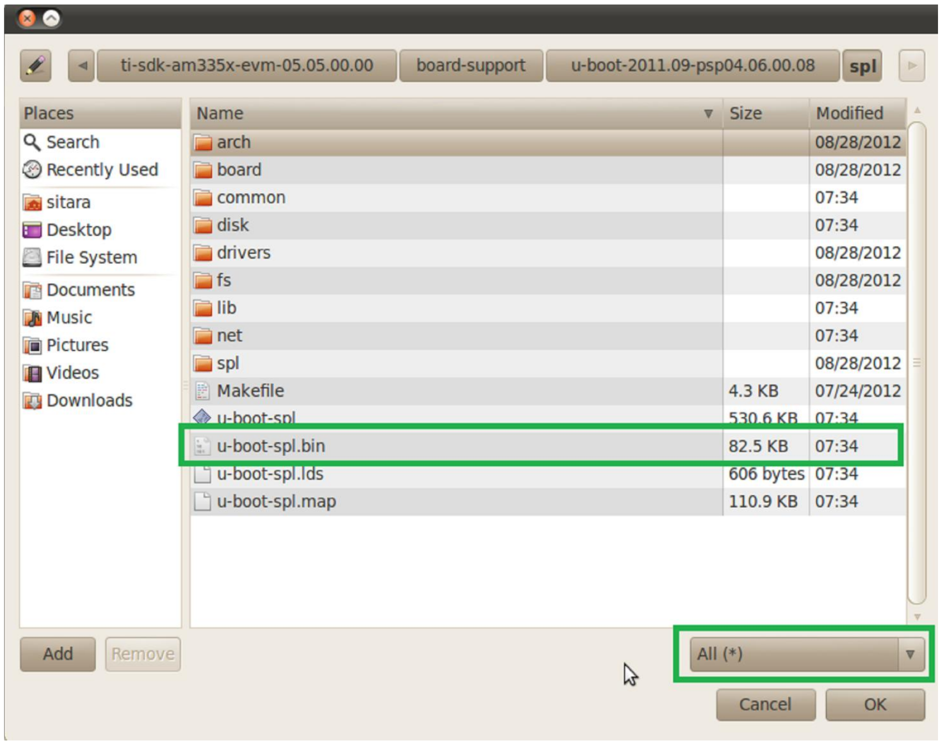
     IMPORTANT

     The *Load Memory* tool is used here because we are loading the SPL binary and not and ELF image. The reason for this is that the binary is sized to be able to fit into the internal RAM of the SoC

   - Click the **Browse** button and browse to the following directory

     /home/sitara/sitara-board-port-uboot/spl

   - Change the filter in the lower-right corner to **All (*)**



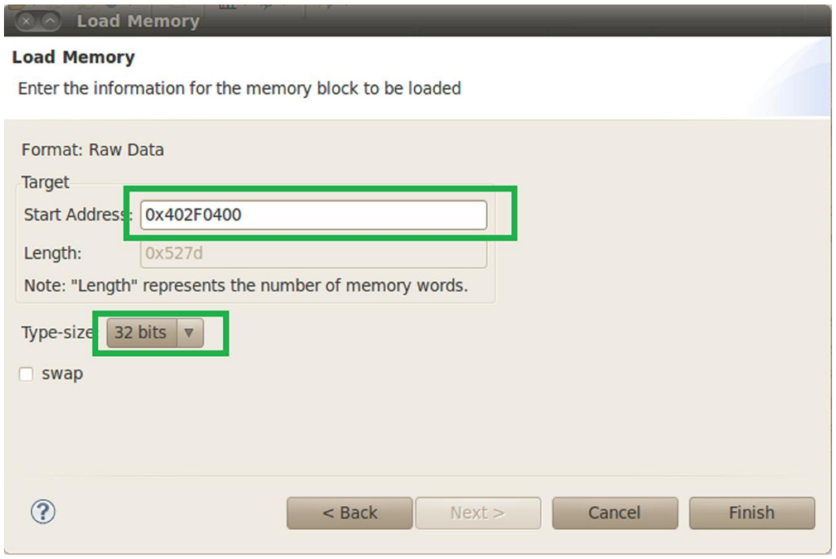
Name	Size	Modified
arch		08/28/2012
board		08/28/2012
common		07:34
disk		07:34
drivers		08/28/2012
fs		08/28/2012
lib		07:34
net		07:34
spl		08/28/2012
Makefile	4.3 KB	07/24/2012
u-boot-spl	530.6 KB	07:34
u-boot-spl.bin	82.5 KB	07:34
u-boot-spl.lds	606 bytes	07:34
u-boot-spl.map	110.9 KB	07:34

- Select the **u-boot-spl.bin** file and click **OK**
- You should now be back at the *Load Memory* dialog. Select **Next**
- In the next screen change the following settings
    - **Start Address: 0x402F0400**

        IMPORTANT

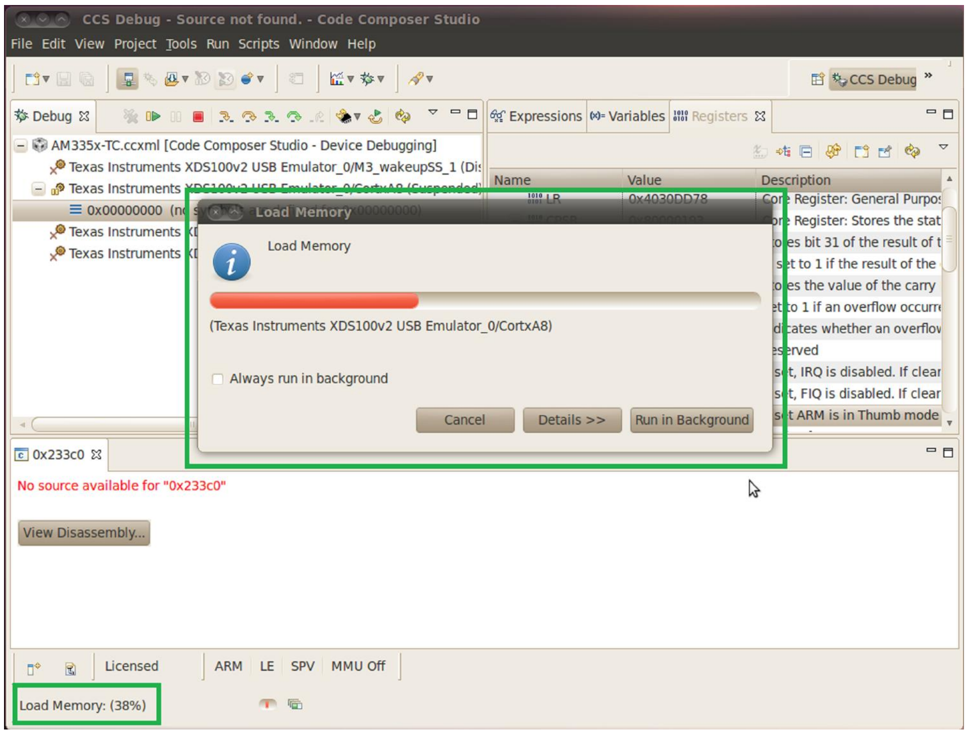
        This is the start address of the SPL binary as defined in the u-boot sources

    - **Type-size: 32 bits**



Format: Raw Data
Target
Start Address: 0x402F0400
Length: 0x527d
Note: "Length" represents the number of memory words.
Type-size 32 bits
swap

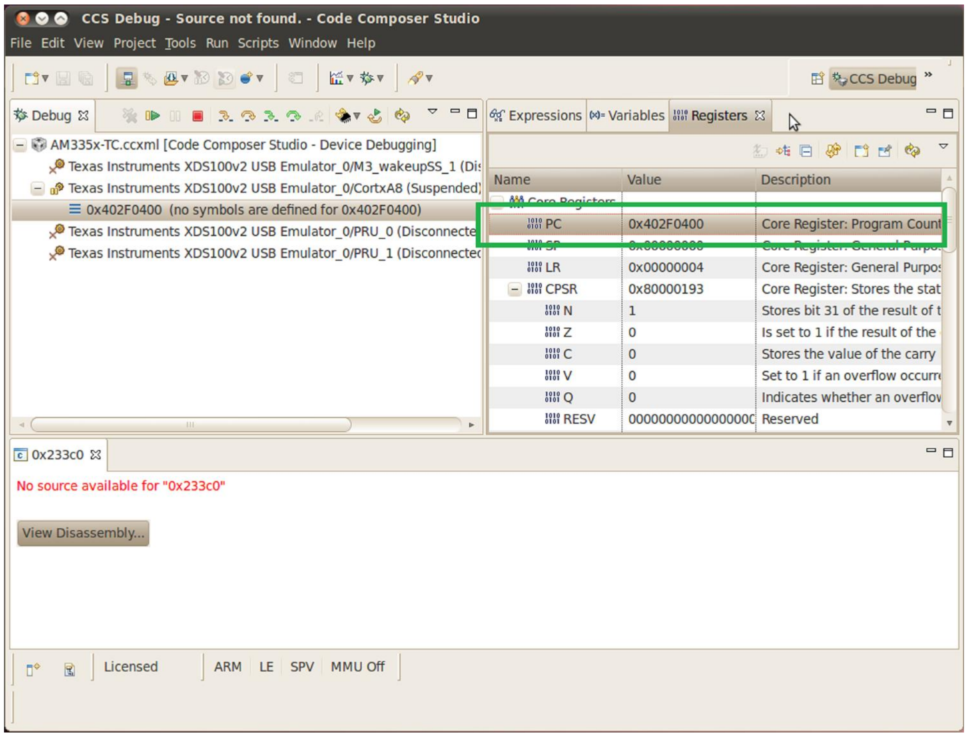
- Click **Finish**
- You should see a box pop up showing the memory load operation


CCS Debug - Source not found. - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Load Memory
Load Memory
(Texas Instruments XDS100v2 USB Emulator_0/CortxA8)
Always run in background
Cancel
Details >>
Run in Background
No source available for "0x233c0"
View Disassembly...
Load Memory: (38%)

NOTE

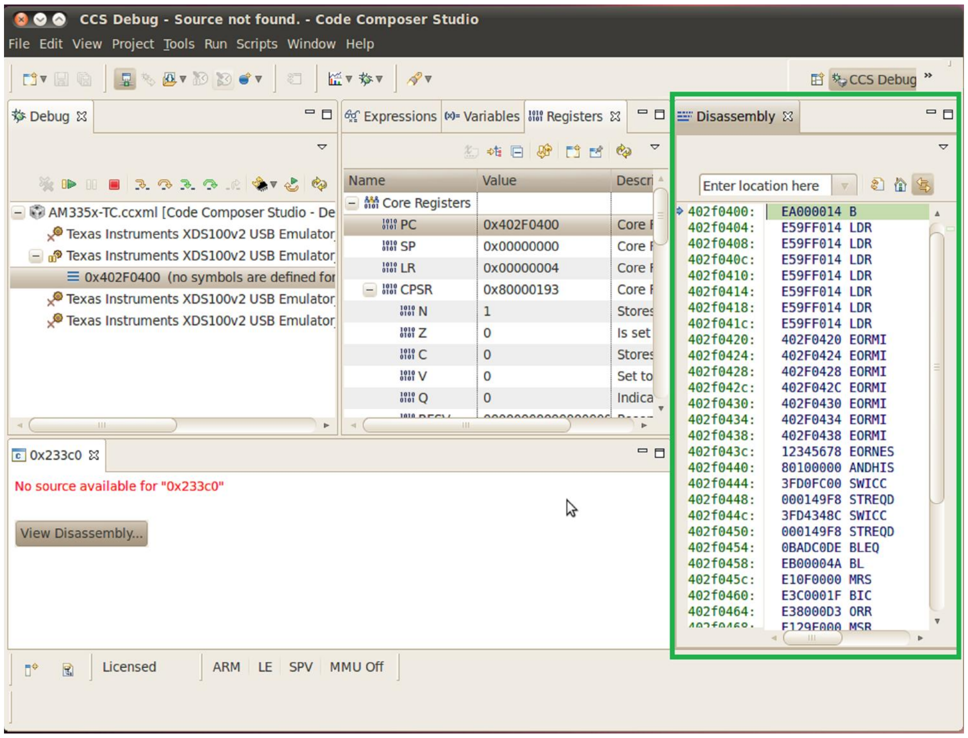
This may take a bit of time depending on the JTAG emulator. Just be patient and allow the process to finish

5. Now that the SPL binary is loaded you can use the following commands to load the symbols for that binary to allow source level debugging
   - Click **Run -> Load -> Load Symbols...**
   - In the dialog box click the **Browse** button
   - Like before browse to the **spl** directory and change the filter to **All**
   - Select the **u-boot-spl** file which is the ELF executable that also contains the symbols and click **OK**
   - Click **OK**
6. Now that the binary and symbols are loaded we need to set the program counter to the beginning of the SPL code. To do this go back to your **Registers** Tab and in the **Core Registers** list change the PC value to the start address you entered before (**0x402F0400**)


CCS Debug - Source not found. - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Debug
Expressions
Variables
Registers
0x402F0400 (no symbols are defined for 0x402F0400)
No source available for "0x233c0"
View Disassembly...

Name	Value	Description
Core Registers		
PC	0x402F0400	Core Register: Program Count
SP	0x00000000	Core Register: General Purpo
LR	0x00000004	Core Register: General Purpos
CPSR	0x80000193	Core Register: Stores the stat
N	1	Stores bit 31 of the result of t
Z	0	Is set to 1 if the result of the
C	0	Stores the value of the carry
V	0	Set to 1 if an overflow occurre
Q	0	Indicates whether an overflow
RESV	0000000000000000	Reserved

7. Click **View -> Disassembly** to open the Disassembly view and see that the program counter is now at *0x402F0400* and that there appears to be valid content in the memory.



CCS Debug - Source not found. - Code Composer Studio

```
AM335x-TC.ccxml [Code Composer Studio - De
  Texas Instruments XDS100v2 USB Emulator
  Texas Instruments XDS100v2 USB Emulator
    0x402F0400  (no symbols are defined for
  Texas Instruments XDS100v2 USB Emulator
  Texas Instruments XDS100v2 USB Emulator
```

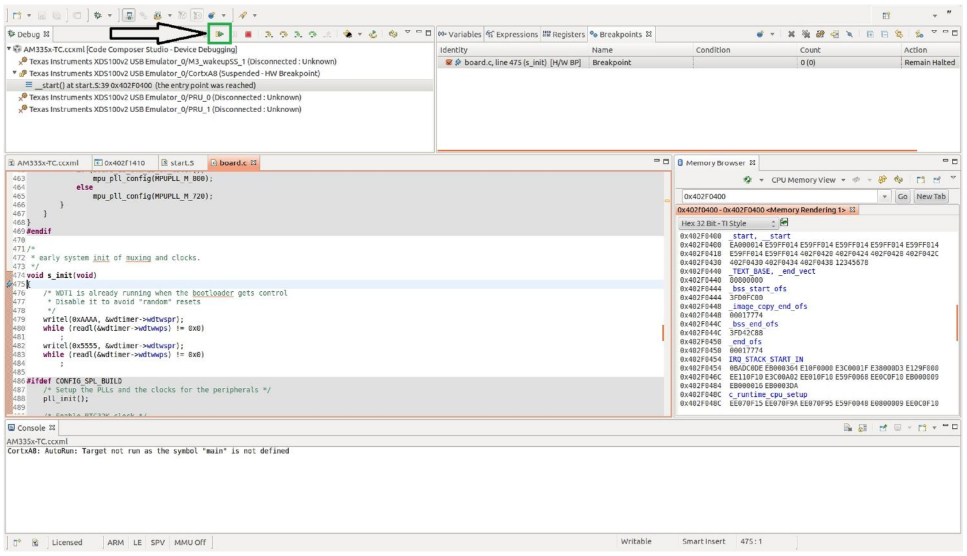
Name	Value	Descri
Core Registers		
PC	0x402F0400	Core
SP	0x00000000	Core
LR	0x00000004	Core
CPSR	0x80000193	Core
N	1	Stores
Z	0	Is set
C	0	Stores
V	0	Set to
Q	0	Indica

```
0x233c0
No source available for "0x233c0"
View Disassembly...
```

```
402f0400:  EA000014 B
402f0404:  E59FF014 LDR
402f0408:  E59FF014 LDR
402f040c:  E59FF014 LDR
402f0410:  E59FF014 LDR
402f0414:  E59FF014 LDR
402f0418:  E59FF014 LDR
402f041c:  E59FF014 LDR
402f0420:  402F0420 EORMI
402f0424:  402F0424 EORMI
402f0428:  402F0428 EORMI
402f042c:  402F042C EORMI
402f0430:  402F0430 EORMI
402f0434:  402F0434 EORMI
402f0438:  402F0438 EORMI
402f043c:  12345678 EORNES
402f0440:  80100000 ANDHIS
402f0444:  3FD0FC00 SWICC
402f0448:  000149F8 STREQD
402f044c:  3FD4348C SWICC
402f0450:  000149F8 STREQD
402f0454:  0BADC0DE BLEQ
402f0458:  EB00004A BL
402f045c:  E10F0000 MRS
402f0460:  E3C0001F BIC
402f0464:  E38000D3 ORR
```

Licensed | ARM | LE | SPV | MMU Off

8. Now open the source file so we can set a breakpoint in the s_init function
   - Click **File -> Open File**
   - Browse to **/home/sitara/sitara-board-port-uboot/board/ti/am335x/**
   - select the **board.c** file and click **OK**
9. You can now click the green run arrow to start execution of the SPL code. After a moment, select the yellow Halt button



```
AM335x-TC.ccxml [Code Composer Studio - Device Debugging]
  Texas Instruments XDS100v2 USB Emulator_0/M3_wakeupSS_1 (Disconnected : Unknown)
  Texas Instruments XDS100v2 USB Emulator_0/CortxA8 (Suspended - HW Breakpoint)
    __start() at start.S:39 0x402F0400  (the entry point was reached)
  Texas Instruments XDS100v2 USB Emulator_0/PRU_0 (Disconnected : Unknown)
  Texas Instruments XDS100v2 USB Emulator_0/PRU_1 (Disconnected : Unknown)
```

Identity	Name	Condition	Count	Action
board.c, line 475 (s_init) [H/W BP]	Breakpoint		0 (0)	Remain Halted

```c
                mpu_pll_config(MPUPLL_M_800);
            else
                mpu_pll_config(MPUPLL_M_720);
        }
    }
}
#endif

/*
 * early system init of muxing and clocks.
 */
void s_init(void)
{
    /* WDT1 is already running when the bootloader gets control
     * Disable it to avoid "random" resets
     */
    writel(0xAAAA, &wdtimer->wdtwspr);
    while (readl(&wdtimer->wdtwwps) != 0x0)
        ;
    writel(0x5555, &wdtimer->wdtwspr);
    while (readl(&wdtimer->wdtwwps) != 0x0)
        ;

#ifdef CONFIG_SPL_BUILD
    /* Setup the PLLs and the clocks for the peripherals */
    pll_init();
```

```
0x402F0400 - 0x402F0400 <Memory Rendering 1>
Hex 32 Bit - TI Style
0x402F0400  _start, __start
0x402F0400  EA000014 E59FF014 E59FF014 E59FF014 E59FF014 E59FF014
0x402F0418  E59FF014 E59FF014 402F0420 402F0424 402F0428 402F042C
0x402F0430  402F0430 402F0434 402F0438 12345678
0x402F0440  _TEXT_BASE, _end_vect
0x402F0440  80800000
0x402F0444  _bss_start_ofs
0x402F0444  3FD0FC00
0x402F0448  _image_copy_end_ofs
0x402F0448  00017774
0x402F044C  _bss_end_ofs
0x402F044C  3FD42C88
0x402F0450  _end_ofs
0x402F0450  00017774
0x402F0454  IRQ_STACK_START_IN
0x402F0454  0BADC0DE EB000364 E10F0000 E3C0001F E38000D3 E129F000
0x402F046C  EE110F10 E3C00A02 EE010F10 E59F0068 EE0C0F10 EB000009
0x402F0484  EB000016 EB0003DA
0x402F048C  c_runtime_cpu_setup
0x402F048C  EE070F15 EE070F9A EE070F95 E59F0048 E0800009 EE0C0F10
```

```
AM335x-TC.ccxml
CortxA8: AutoRun: Target not run as the symbol "main" is not defined
```

Licensed | ARM | LE | SPV | MMU Off | Writable | Smart Insert | 475 : 1

   - You should see the instruction pointer on the while(myVar); instruction. As discussed earlier this was created to prevent the SPL code from running any further than beyond configuring the EMIF to communicate with the DDR3
10. Once you are finished testing this go back to your terminal and navigate to the **sitara-board-port-uboot directory**. Because the next lab uses BeagleBone and we modified the template for the BeagleBoneBlack we want to reset the changes we just made to board.c and ddr_defs.h.

    To reset the changes type in **git reset --hard**

Archived Versions

- **05.07.00.00 (http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Training:_Tuning_the_DDR3_Timings_on_BeagleBoneBlack&oldid=154464)**

Engage in the TI E2E Community
Ask questions, share knowledge, explore ideas and help solve problems with fellow engineers

For technical support please post your questions at http://e2e.ti.com/. Please post only comments about the article ***Sitara Linux Training: Tuning the DDR3 Timings on BeagleBoneBlack*** *here.*

Links

Amplifiers & Linear (http://www.ti.com/lsds/ti/analog/amplifier_and_linear.page)
Audio (http://www.ti.com/lsds/ti/analog/audio/audio_overview.page)
Broadband RF/IF & Digital Radio (http://www.ti.com/lsds/ti/analog/rfif.page)
Clocks & Timers (http://www.ti.com/lsds/ti/analog/clocksandtimers/clocks_and_timers.page)
Data Converters (http://www.ti.com/lsds/ti/analog/dataconverters/data_converter.page)

DLP & MEMS (http://www.ti.com/lsds/ti/analog/mems/mems.page)
High-Reliability (http://www.ti.com/lsds/ti/analog/high_reliability.page)
Interface (http://www.ti.com/lsds/ti/analog/interface/interface.page)
Logic (http://www.ti.com/lsds/ti/logic/home_overview.page)
Power Management (http://www.ti.com/lsds/ti/analog/powermanagement/power_portal.page)

Proce
(http:

-
-
-
-

Retrieved from "http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Training:_Tuning_the_DDR3_Timings_on_BeagleBoneBlack&oldid=156955"

Categories: AM437x | AM335x | AM35x | AM37x | AM1x | AM18x

- This page was last modified on 17 July 2013, at 15:48.
- This page has been accessed 3,874 times.
-