Speeding AM335x Programmable Real-time Unit (PRU) Application Development Through Improved Debug Tools

The hardware modules and descriptions referred to in this document are ***NOT SUPPORTED*** by Texas Instruments (www.ti.com / e2e.ti.com).

These materials are intended for do-it-yourself (DIY) users who want to use the PRU at their own risk without TI support. "Community" support is offered at BeagleBoard.org/discuss.

Agenda

- Introduction to the PRU Subsystem (PRU-ICSS on AM335x)
- Chip-level and Subsystem Level over of the PRU-ICSS
- PRU debug capabilities now included in CCS with details on how to use them
- PRU Resources

Introduction to the PRU SubSystem

- What is PRU SubSystem?
  - **P**rogrammable **R**eal-time **U**nit **S**ub**S**ystem
  - Dual 32bit RISC processors
  - Local instruction and data RAM; access to SoC resources.

- What devices include PRU SubSystem?
  - Legacy PRUSS: OMAPL137/ AM17x, OMAPL138/ AM18x, C674x
  - PRU-ICSS* (PRUSSv2): AM335x

- Why PRU SubSystem?
  - Full programmability allows adding customer differentiation
  - Efficient in performing embedded tasks that require manipulation of packed memory mapped data structures
  - Efficient in handling of system events that have tight real-time constraints.

* PRU-ICSS = **P**rogrammable **R**eal-time **U**nit and **I**ndustrial **C**ommunication **S**ub**S**ystem.

PRU Subsystem Is / Is-Not

IS	IS-Not
Dual 32-bit RISC processor specifically designed for manipulation of packed memory mapped data structures and implementing system features that have tight real time constraints	In not a H/W accelerator to speed up **algorithm computations**.
Simple RISC ISA - Approximately 40 instructions - Logical, arithmetic, and flow control ops all complete in a single cycle	Is **not a general purpose RISC processor** - No multiply hardware/instructions - No cache - No pipeline - No C programming
Could be used to enhance the existing peripheral feature set or implement new peripheral capability with software bit bang	Is not a stand alone configurable peripheral and will need some hardware assist for configurable peripheral implementation
Includes example code to demonstrate various features. Examples can be used as building blocks.	No Operating System or high level application software stack

PRU Value

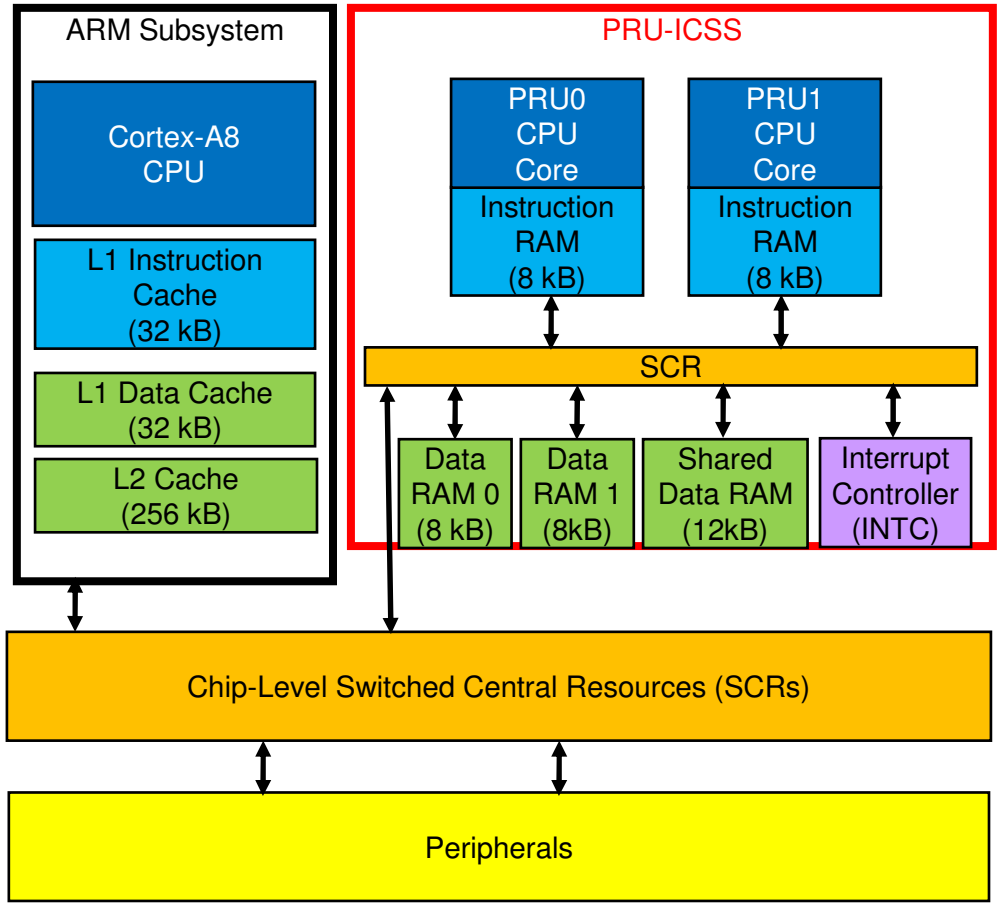
- Extend connectivity and peripheral capability
  - Implement Industrial Communications protocols (like EtherCAT®,PROFINET, EtherNet/IP™, PROFIBUS, POWERLINK, SERCOS III)
  - Implement special peripherals and bus interfaces (like soft UARTs interfaces)
  - Digital IOs with latency in ns
  - Implement smart data movement schemes (especially useful for audio algorithms like reverb, room correction, etc.)

- Reduce system power consumption
  - Allows switching off both ARM and DSP clocks
  - Implement smart power controller by evaluating events before waking up DSP and/or ARM. Maximized power down time.

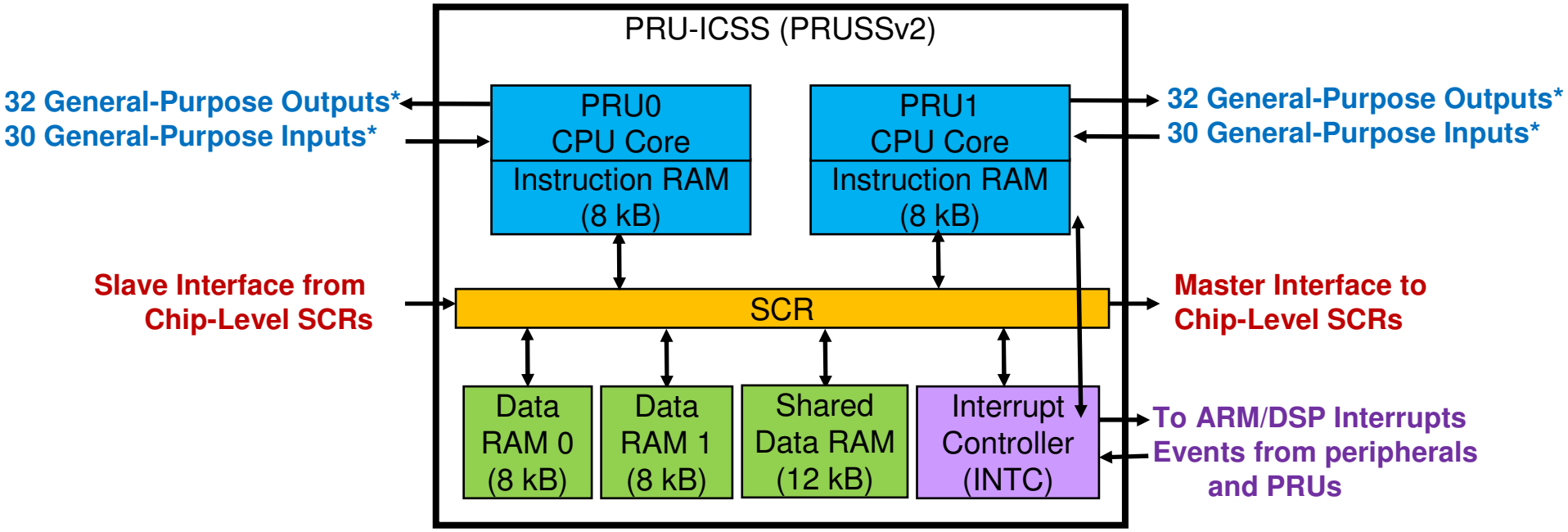
- Accelerate system performance
  - Full programmability allows custom interface implementation
  - Specialized custom data handling to offload CPU

Chip-level Integration of the PRU-ICSS (PRUSSv2)

- ARM has access to PRU-ICSS memory
- PRU-ICSS has access to its own local memories and other chip-level memory resources and peripherals



PRU-ICSS (PRUSSv2) Block Diagram



* On AM335x, only 15 General-Purpose Outputs and 16 General-Purpose Inputs are pinned out.

PRU Development Support Integrated in CCS

		CCS 5.x
pasm (PRU assembler)		No
AM335x PRU Debug Tools	Disassembly window	Yes
	Memory windows	Yes
	Register windows	Yes
	Execution controls	Yes
	Soft reset control	Yes
	Sleep control	Yes

Download the CCS 5.x here:

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

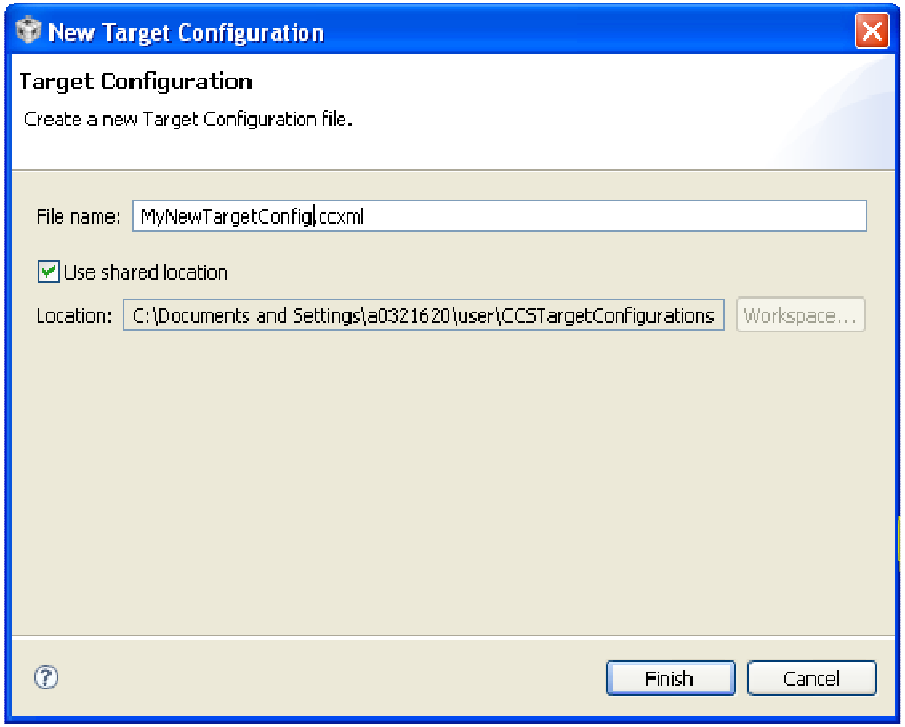
Summary of PRU Debug Capabilities

- Disassembly window to show PRU assembly code
- Memory windows to show PRU program and data memory contents
  - Ability to load/fill memory contents
  - Ability to save memory contents
  - Ability to load PRU code binaries
- Register windows to show PRU subsystem control, data and status registers
  - View and modify PRU subsystem registers
- Execution controls
  - Run/Halt
  - Single-stepping through assembly instructions
  - Breakpoint control
- PRU soft reset control

How to create a target configuration [1]

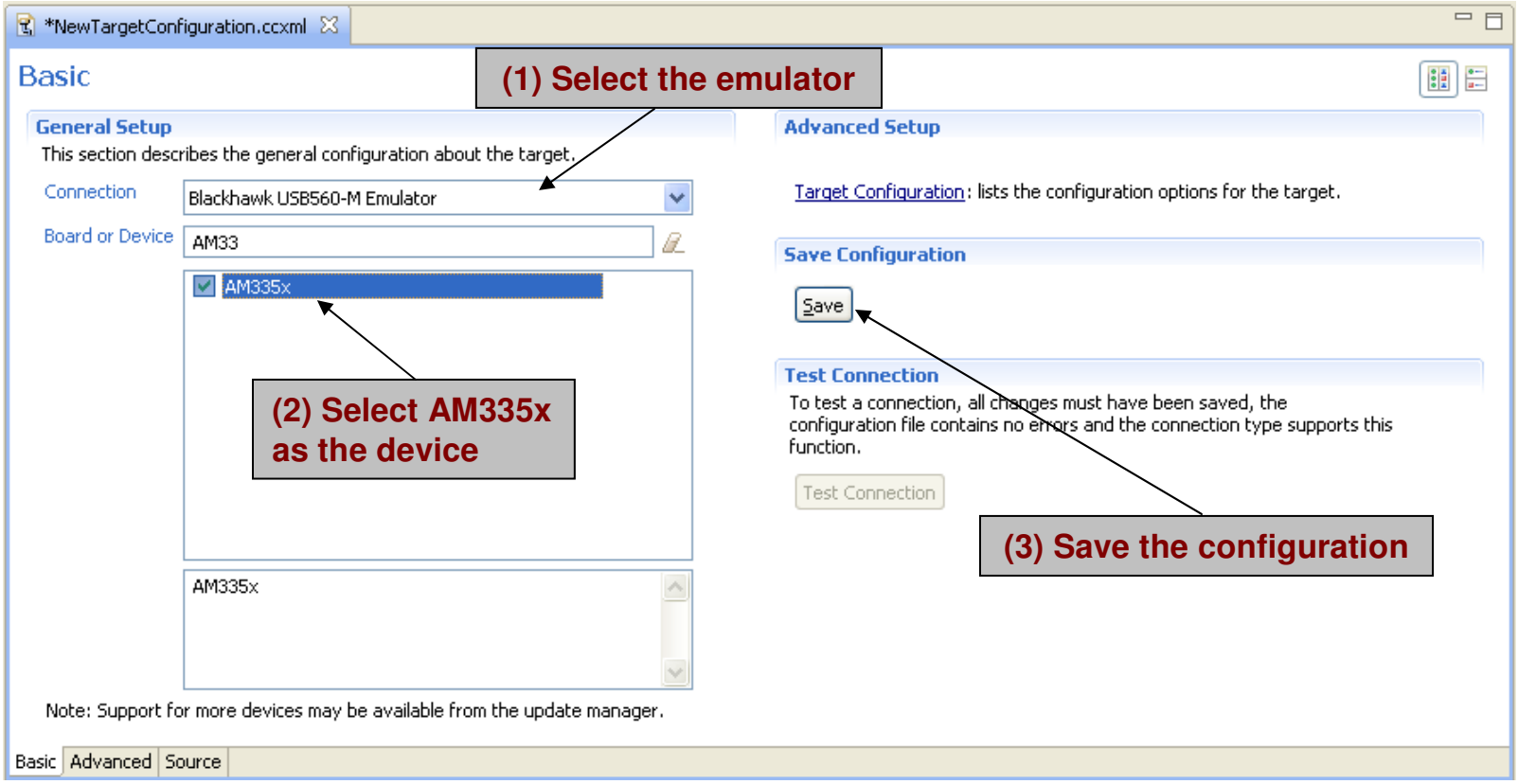
- You must choose the correct target configuration for CCS to include the PRU debug capabilities

- To create a new target configuration:
  - Open CCS
  - Click on Target > New Target Configuration…
  - CCS will open a New Target Configuration window, in this window:
    - Type the desired filename for the target configuration (like MyTargetConfig.ccxml)
    - If the storage location is correct, the click Finish.



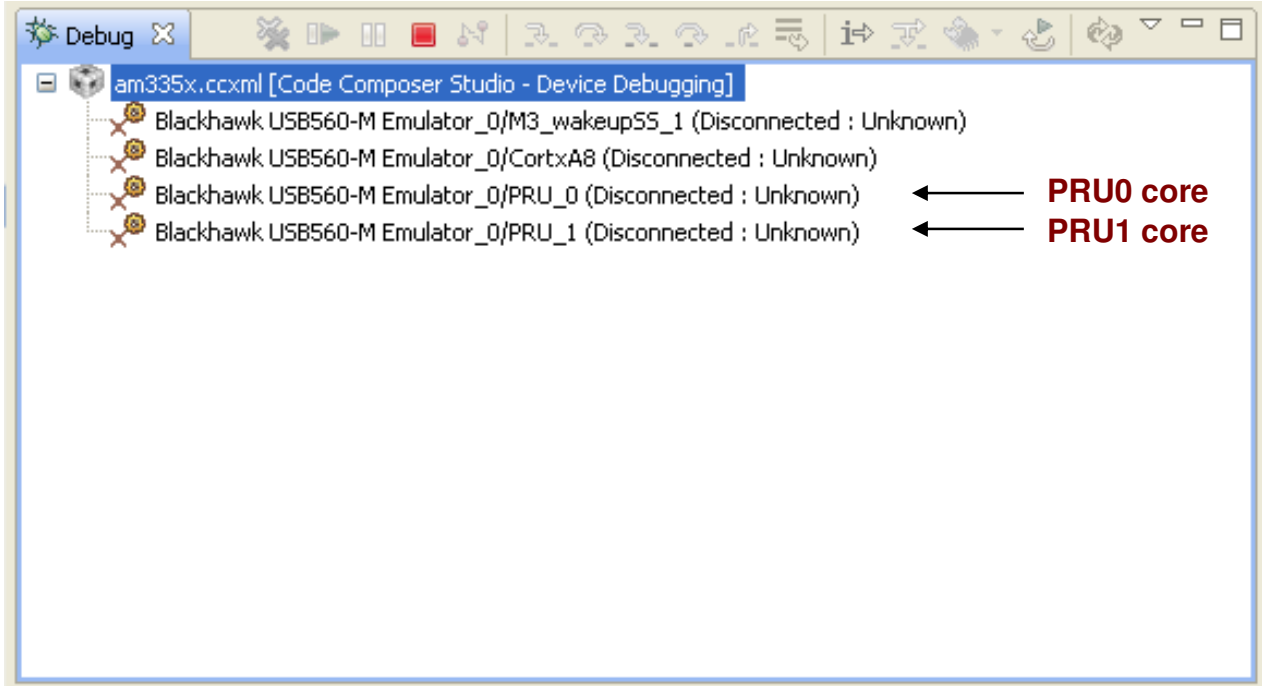
How to create a target configuration [2]

- In the main screen, CCS will open a “New Target Configuration” tab
  - In the Connection menu, choose the correct emulator you plan to use. The debug tools do not apply to the simulator.
  - In the Device box, choose AM335x from the options.
  - Click on the Save button under Save Configuration.



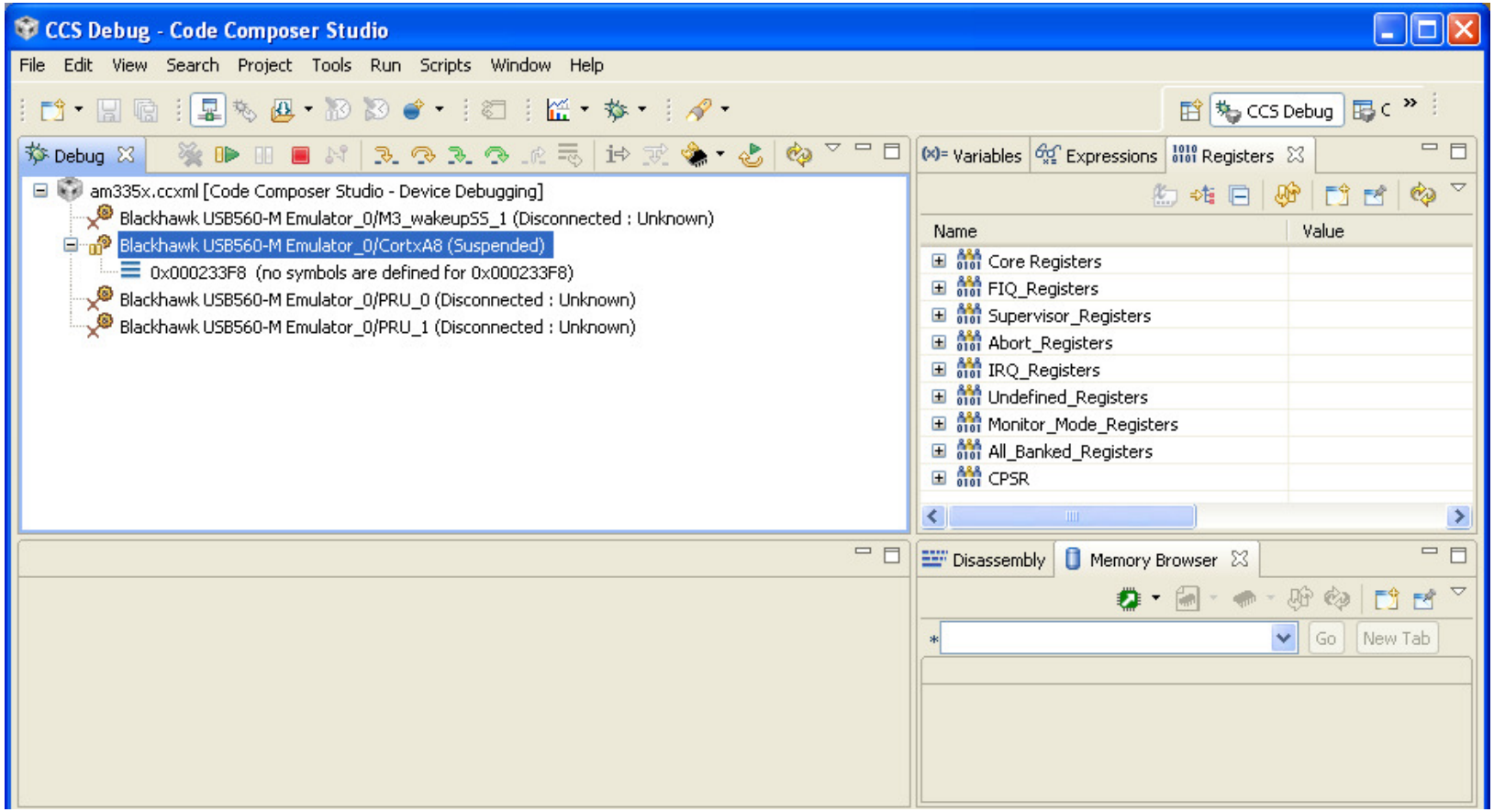
How to connect to a target PRU

- Start a debug session:
  - Click on Target > Launch TI Debugger or click on the
  - CCS will connect to the target
- The Debug window will now show 4 disconnected targets:
  - **The M3_wakeupSS_1 core**
  - **The CortexA8 core**
  - **The PRU0 core**
  - **The PRU1 core**



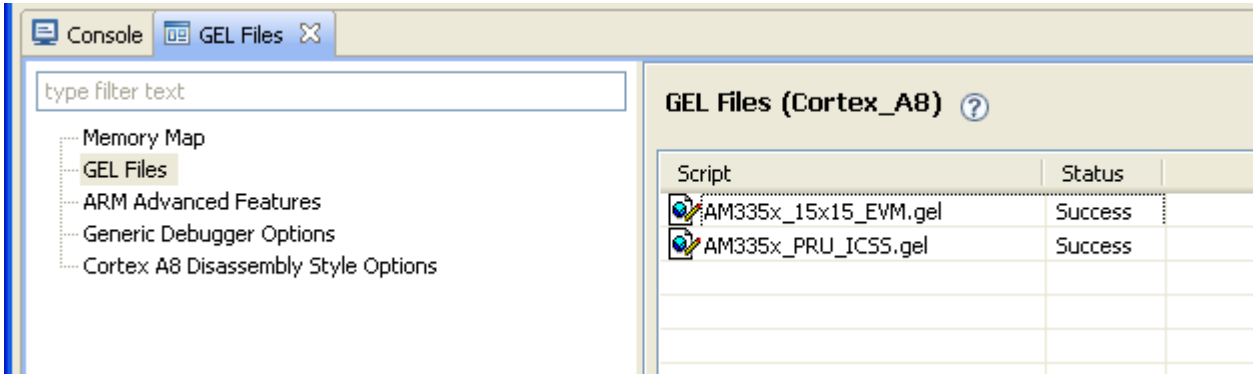
How to connect to a target PRU

- Select Disconnected Device on the CortexA8 and either:
  - Right-click and then select Connect Target, or
  - Use Target > Connect Target from the Menu on top
- CCS will connect to the CortexA8 and the Register, Disassembly and Memory windows will populate.

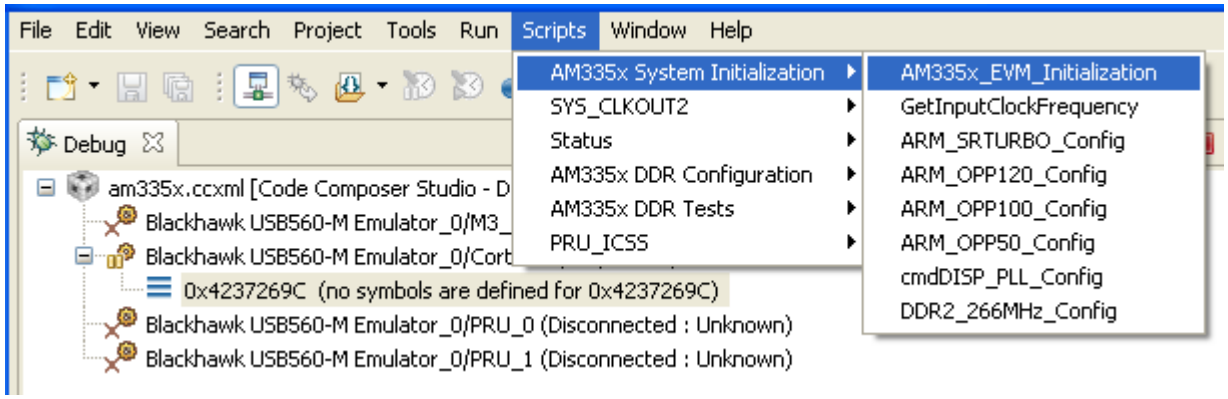


How to connect to a target PRU

- Load AM335x CortexA8 and PRU gel files
  - Tools → GEL Files
  - Right click in GEL Files box and load AM335x_15x15_EVM.gel
  - Right click in GEL Files box and load AM335x_PRU_ICSS.gel

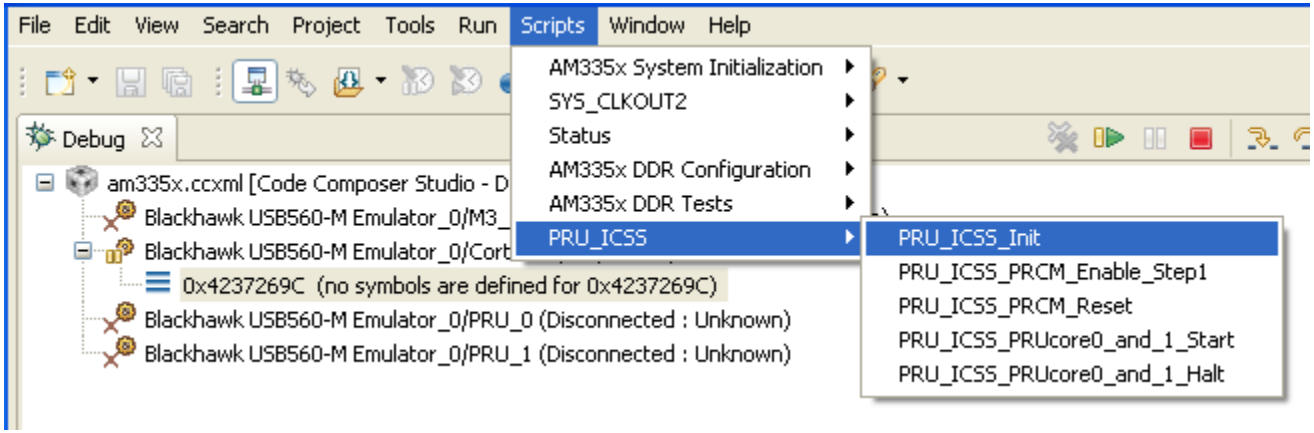


- Run the AM335x System Initialization script

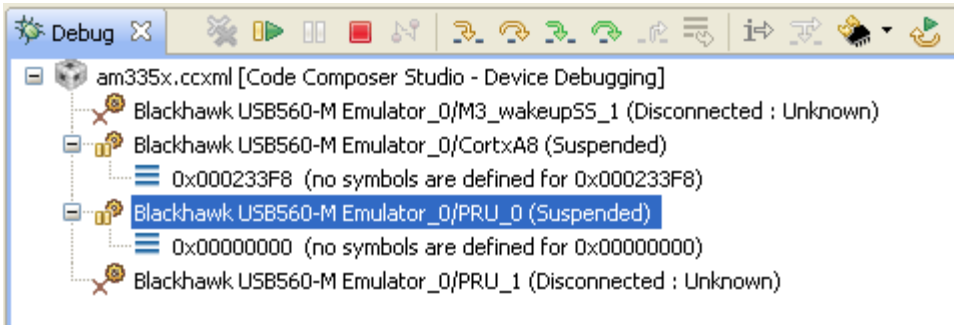


How to connect to a target PRU

- Run the PRU_ICSS_Init script

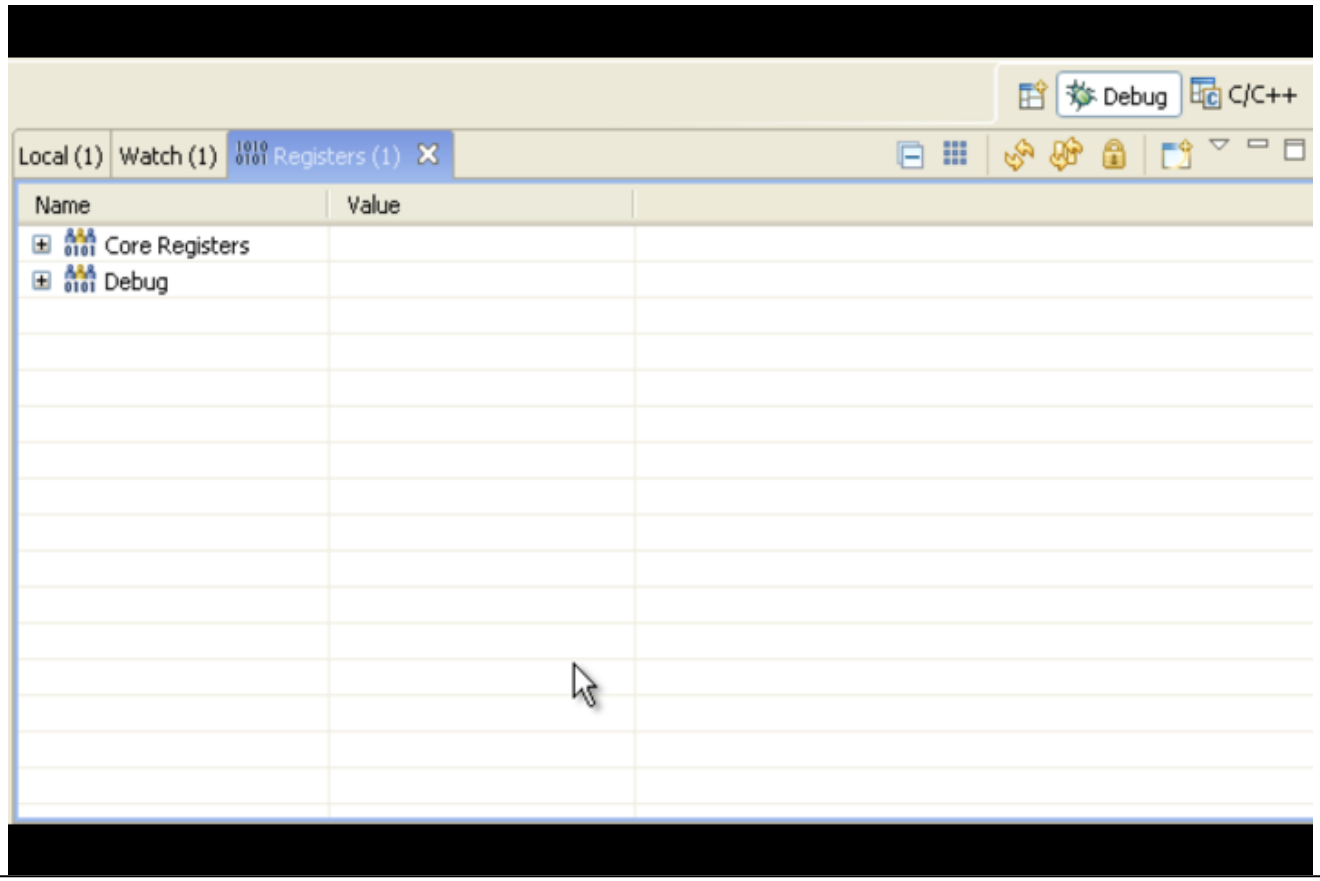


- Select Disconnected Device on the PRU core and either:
  - Right-click and then select Connect Target, or
  - Use Target > Connect Target from the Menu on top
- CCS will connect to the PRU core and the Register, Disassembly and Memory windows will populate.



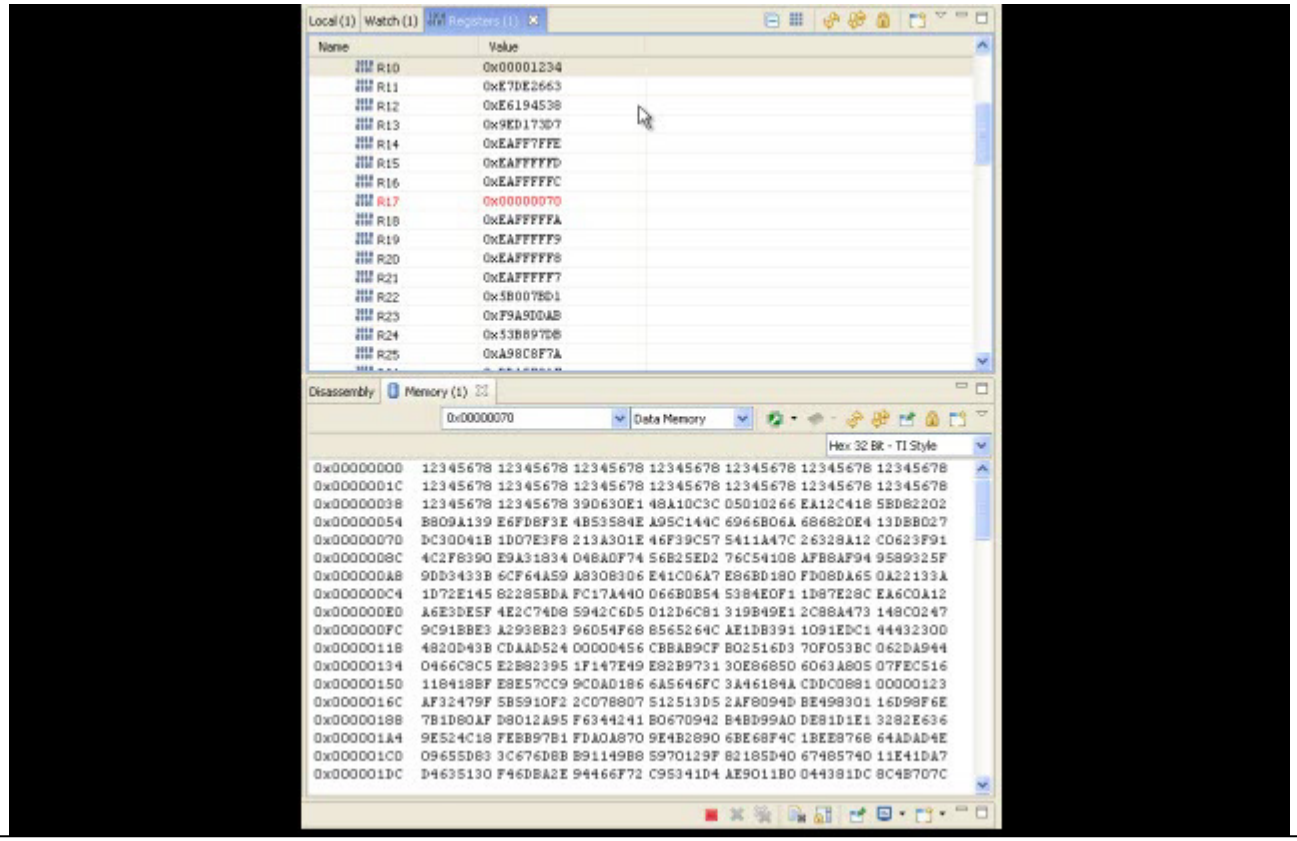
The Register Window

- When one of the connected PRUs is selected in the Debug Window, the Registers Window will show the Core Registers and Debug Registers for that PRU. You can expand and hide the Core and Debug registers in groups.
- You can modify values directly in the register window by double-clicking on the value and entering the new value.



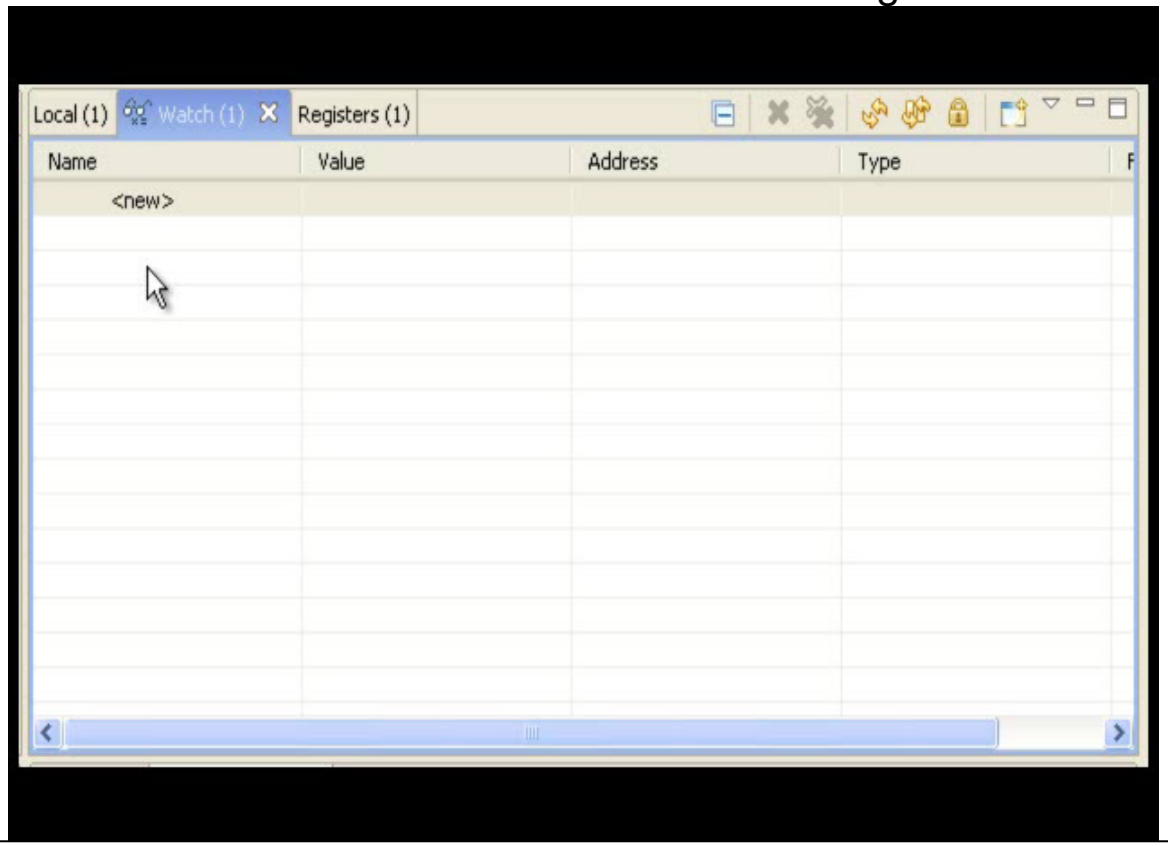
The Register Window

- You can choose a register in the Register window and go to the memory address in the register:
  - Select the desired register
  - Right-click and select Open Memory View at Register Value
  - The Memory Window will go to the address specific in the selected register



The Register Window

- You can also choose a register to be a watch value in the Watch window:
  - Select the desired register
  - Right-click and select Create Watch Expression
  - The selected register is now a watch value in the Watch Window
  - You can put watch values from multiple CPUs in the same window, but CCS will only display the contents for the CPU selected in the debug window.



The Memory Window

Address Selection:
Enter text or use menu
for recently used values

Memory Selection:
- Program
- Data
- PRU Device Memory

Save, Load or
Fill Memory

Data Display Style

Disassembly | Memory (1)

0x0 | Program Memory | Hex 32 Bit - TI Style

```
0x00000000  9100A387 5100E90A 24002000 FF00C70E EF00C80E 0400E9E9 0000E7E7 0000E8E8
0x00000008  8100A387 5100E902 7F000 F8 2A000000 848449CC 0849BAE6 92A5F307 A0281136
0x00000010  EBF5C530 C31A9750 6E4224 5A7F748F 0D15AC38 8E8C0867 33449409 164AA838
0x00000018  AA9E252E 7A142E3D 809999ED E193DB 092C108B 57550387 65F04210 C9B2E0C0
0x00000020  ACAE61A3 9FC65D14 40F6C917 A 2B2 E33DAE00 C56E2A72 CE829C0F 8E81034C
0x00000028  CF429191 168A066A E264F05B 23 27ECB0C8 9860EDA6 5AAE0250 0D96083C
0x00000030  0C4C9795 7D5112C1 ADF7A408 83D 37654019 55F68E5F
0x00000038  023AC50C 7F895C87 A5824E2E 332 6E40F314 0AA02E7D
0x00000040  76847893 1310F66C D1043522 AE2 9B4CB06E 16F299C4
0x00000048  402826C5 7328274B C0F0C090 CE6 FB42A454 AFACA002
0x00000050  0CF2C833 F06A3E60 E542C466 4A2 53C685AA 21A07010
0x00000058  3228A4C1 6F083730 C0FCD4AD 690 8468F42D 45E24039
0x00000060  4DC4530F B133F5AE 90481F5B 88911201 BB84B073 57B2652D 60D2A15C 60408865
0x00000068  72BA6206 47A19500 25328C1D 0BE40C60 6A86038D 7CEE3C6A C08C8741 6BDEBB36
0x00000070  E015C16C 91384C93 D21F50EB 0484D669 A5B0013A 81A03484 F80809D1 60572AE5
0x00000078  A9F31F6A 862B18F3 28B01423 BCB14F0A CA999CAA 3A183571 88E71D25 73A65F20
0x00000080  32C52C92 D1D92092 504B0717 D0EBE0B0 4DA92424 77064208 6C13118D 08D169BE
0x00000088  E36BD47B E130E982 0E680C48 7456B185 D892A3CD 938F444E 10C20BEA 45D2D085
```

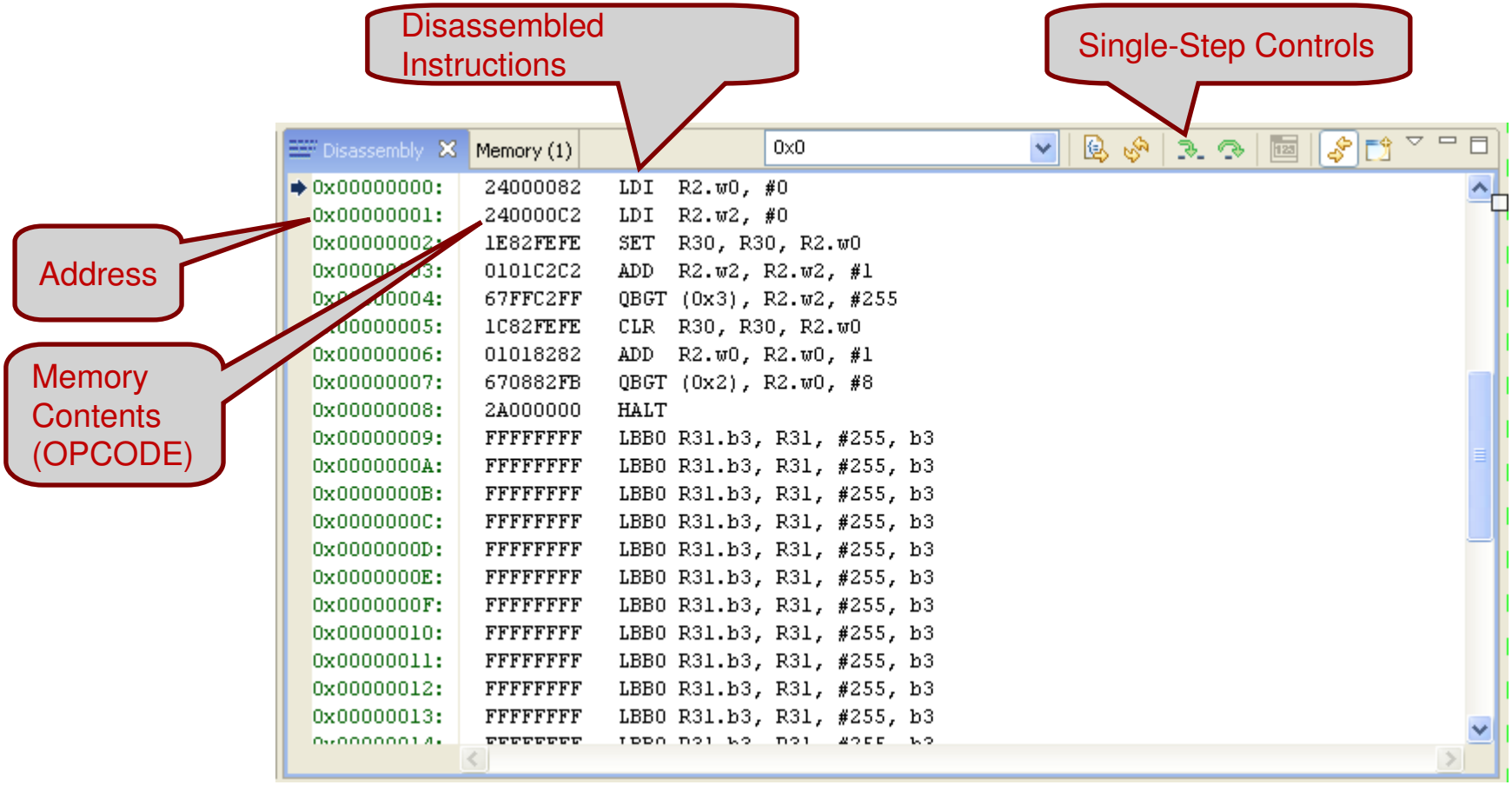
Double-click on a
memory location
to modify it
directly

Loading a PRU Binary (executable file)

- To load a PRU binary, it must be loaded through the Memory Window (unlike other CPUs in CCS).
  - Go to the Memory Window, pull down the menu for 
  - Select Load, select the binary file you want to load, then click on Next
  - In the next window…
    - Enter the Start Address (usually 0x0)[see note 1]
    - Choose Program Memory as the Memory Page
    - Choose 32-bits as the Type Size
    - Click Finish
- The binary is now loaded and if you go to the Disassembly window, you will see the code.

[Note 1] The PRU assembler always assembles code to start at program address 0x0. But you could do code overlays by modifying code to start at another address. In this case, you would use some address other than 0x0 for the binary being loaded.

The Disassembly Window



- Both single-step controls do the step through a single assembly instruction
- Not all of the controls in the disassembly window have a function on the PRU

Execution Control – Run, Halt, Single-Step

- To Run/Halt, with the PRU selected
  - Click the RUN button in the Debug window, or
  - Click Target > Run

- The PRU code will run until it is halted or executes a HALT instruction

- To single-step through code, use any of the single step controls.
  - Step-Into, Step-Over, Assembly Step-Into, Assembly Step-Over all have identical functions on the PRU

Execution Control – Breakpoint Control

- You can set/clear breakpoints by double-clicking in the bar to the left of the address in the disassembly window.

- You can also view/control PRU breakpoints by using View > Breakpoints

Execution Control Screenshot

CCS Debug - Code Composer Studio

File Edit View Search Project Tools Run Scripts Window Help

Debug

am335x.ccxml [Code Composer Studio - Device Debugging]
- Blackhawk USB560-M Emulator_0/M3_wakeupSS_1 (Disconnected : Unknown)
- Blackhawk USB560-M Emulator_0/CortxA8 (Suspended)
  - 0x000233F8 (no symbols are defined for 0x000233F8)
- Blackhawk USB560-M Emulator_0/PRU_0 (Suspended)
  - 0x00000002 (no symbols are defined for 0x00000002)
- Blackhawk USB560-M Emulator_0/PRU_1 (Disconnected : Unknown)

Registers

Name	Value	Description
Core Registers		
PC	0x00000002	Core Register: PCOU
R0	0x01C14100	Core Register: R0 Re
R1	0x41D0CC5A	Core Register: R1 Re
R2	0x5BCBB777	Core Register: R2 Re
R3	0x697D84EA	Core Register: R3 Re
R4	0xD68CD451	Core Register: R4 Re

Breakpoints

Identity	Name	Condition	Count	Action
0x0000002	Breakpoint		0 (0)	Remain Halted
0x0000004	Breakpoint		0 (0)	Remain Halted

Disassembly | Memory Browser

Enter location here

```
00000000:  24410080  LDI  R0.w0, #1664
00000001:  2401C1C0  LDI  R0.w2, #449
00000002:  F1542081  LBBO R1.b0, R0, #
00000003:  81002381  SBCO R1.b0, C3, #
00000004:  11F02121  AND  R1.b1, R1.b1
00000005:  13012121  OR   R1.b1, R1.b1
```

Console | GEL Files

am335x.ccxml

```
CortxA8: Output:        **** PRU-ICSS PRCM Enable Step 1 is Done ****
CortxA8: Output:        **** PRU-ICSS PRCM Enable Step 2 is in progress ****
CortxA8: Output:        **** PRU-ICSS PRCM Enable Step 2 is Done ****
CortxA8: Output:        **** PRU-ICSS PRCM Reset is in progress ****
CortxA8: Output:        **** PRU-ICSS PRCM Reset is Done ****
```

Licensed

PRU-ICSS Documentation and Resources

- CCS External Download at
  http://processors.wiki.ti.com/index.php/Download_CCS
- AM335x PRU-ICSS Documentation is available in the
  AM335x PRU-ICSS package