Debug and Trace Capabilities on OMAPL1X/AM1x/DA8x devices

Contents

Debug and Trace capabilities on OMAPL1x/DA8x/AM1x

Hardware setup

Software Setup

Unified BreakPoint Manager (UBM)

Enhanced Trace buffer (ETB)
- Connecting to Target using CCSv5
- Configuring ETB
- Enabling Trace
- Types of Trace settings
- Viewing Trace Data
- ETB - DSP

SYS/BIOS Debugging options
- Real Time analysis(RTA) agent
- RTOS Object View(ROV)
- Exception handling

Linux Debugging using CCSv4/v5

Other useful debugging resources
- Debugging GEL file for boot/application crash related issues
- Exception handling in non-OS environment

Useful Training links/Presentations

Debug and Trace capabilities on OMAPL1x/DA8x/AM1x

What is supported:

- ARM9 has a 4 KB Enhanced Trace Buffer (ETB) based trace and some AET based debug using Unified Break point Manager (UBM)
- DSP has limited debug capability using UBM
- BIOS Real time Analysis tools (RTA) and RTOS Object View(ROV) tools.

What is not supported:

- Directing trace data from the C674x DSP to ETB is not supported by this device.
- System trace (software instrumentation)
- System events monitoring
- System throughput/latency analysis
- System bus watch point

Hardware setup

- OMAPL13x/ DA8xx / AM1x development platforms
- XDS560 USB emulator

Software Setup

- Code Composer Studio v4/v5 (http://processors.wiki.ti.com/index.php/Download_CCS?DCMP=dsp-mc-opemmp-120828&HQS=dsp-mc-opemmp-pr-sw3)

Unified BreakPoint Manager (UBM)

The UBM is where all breakpoints and advanced debug features such as Advanced Event Triggering are made available.

- SW Breakpoints
- Hardware Breakpoints
- Hardware Watchpoints
- Counters
- Trace

For more details refer to the Unified Breakpoint Manager (http://processors.wiki.ti.com/index.php/Unified_Breakpoint_Manager) wiki.

Examples coming soon !!

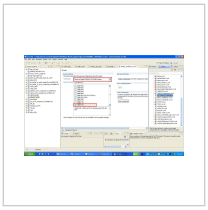
Enhanced Trace buffer (ETB)

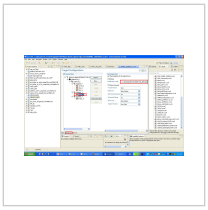
Connecting to Target using CCSv5

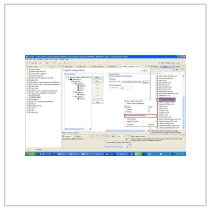
1. Launch CCSv5 IDE and compile the projects that you want to debug
2. Click on "File" Tab and select "New" -> Target Configuration
3. Select the XDS560 Emulator that you are using and pick the target from the list. For Eg in the following image we pick Spectrum digital XDS560USB emulator and use OMAPL138 as the target device. Name the target configuration appropriately.
4. If you want the GEL file to run by default when you connect to the target, goto Advanced Settings, select the core and select the GEL file to be loaded as shown below. Save the configuration when you are done.

Note: GEL file for the platform is found the development platform can be obtained from the board support packages.

1. Go to "View" Tab and select Target configuration. This will launch a Target configuration view where you will see all the available target configurations that you have created. Select the Target configuration that you created in step 3 and 4 and right click and select Launch Target configurations.


Target configuration

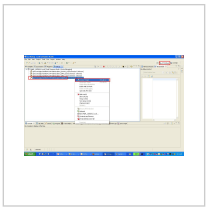

ARM_gel.JPG

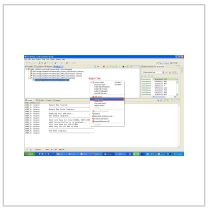

Launch_target.JPG

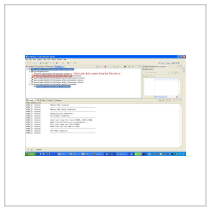
This procedure has been described in detail the wiki section: debug session for OMAPL13x/DA8xx/AM1x device in CCS (http://processors.wiki.ti.com/index.php/How_to_connect_to_the_OMAP-L138/C6748/AM1808_EVM_board_using_CCS%3F#Connecting_using_CCS4)

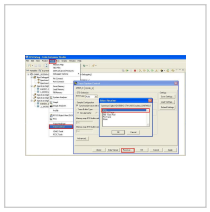
Configuring ETB

1. Connect to the ARM9
2. Right click on the debug session in the Debug View and select "Show all cores"
3. Expand the Non Debuggable Devices node
4. Select the ETB (ETB11_0) and click the connect button.
5. Select the ARM9 in the Debug View
6. Go Tools -> Trace Control
7. Click on the "Receiver..." button at the bottom of the dialog.
8. Select ETB as the Receiver and click OK. Note that there may be a pause of 10-20 seconds before the properties show up in the Trace Control Dialog.
9. Click OK again to exit the trace control dialog.


Connect to ARM9

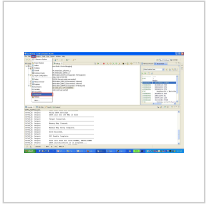

Show_all_cores.JPG

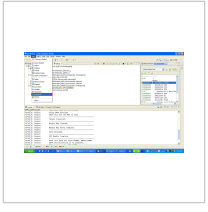

ETB_conect.JPG

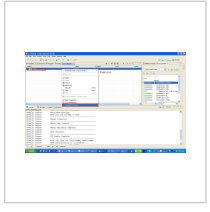

ETB_reciever_select.JPG

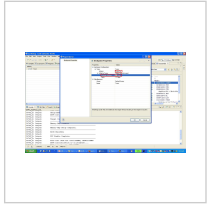
Enabling Trace

1. On the toolbar click on the down arrow beside the breakpoint button (blue circle) and select Trace
2. This will open the breakpoints view and there will be a Trace breakpoint listed. Right click on it and select Breakpoint properties.
3. Under Hardware Configuration, expand Type and What to Trace. Select what you want to capture. Program Address is typical. Then click Ok.
4. Check the box beside the breakpoint to enable it. The color should change from grey to blue.
5. You can now run your application and collect data.


Launch UBM

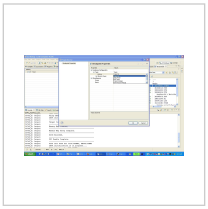

Select_trace_in_UBM.JPG


Trace Properties

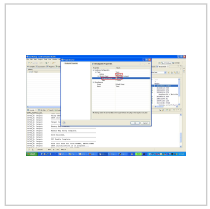

Trace on

Types of Trace settings

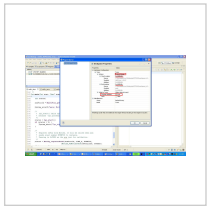
1. **Trace on** : Will use 4KB ETB buffer as circular buffer. Suited for small applications or in case you want to analyze trace leading upto an application failure
2. **Start Trace and End Trace**: To Start and end Trace at custom locations of instruction memory. Useful to analyze/localize application fails upto/after a specific event. For Eg Trace after interrupt received or after ISR execution.
3. **In Range & Don`t trace in range** : Provides Trace in a range of instruction memory. suited to help localize bugs to specific portion of your code.



Trace_type



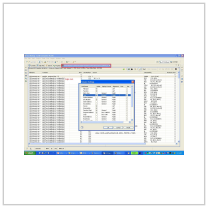
Trace on



Trace in Range

Viewing Trace Data

1. Go to Tools -> Open Trace Connection in New View -> <select the core you enabled trace on>
2. The trace display should now open.



Custom_TraceView

ETB - DSP

Directing trace data from the C674x DSP to ETB is not supported by this device.

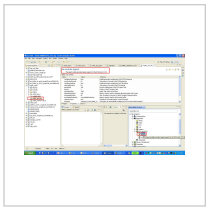
SYS/BIOS Debugging options

- Real Time analysis(RTA) agent
- RTOS Object View(ROV)
- Exception handling

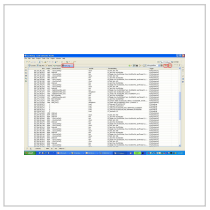
Real Time analysis(RTA) agent

Note: Before Debugging the SYSBIOS Debugging options. It is recommended to set the Preferences under the Windows Tab in CCSv4/v5 to select the version of SYSBIOS, XDCTOOLS in Windows->Preferences->General->RTSC Options. Additionally set IPC and XDAIS if your application uses the components.

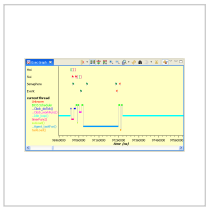
1. Real time analysis can easily be turned on in a SYSBIOS based CCS project using the Grace tools. Select the configuration file (.cfg) in your project this will open an available Resources view inside SYSBIOS. Under diagnostics, select the RTA agent and enable it and save the configuration. Rebuild the project with the new settings.
2. To run your project, choose **Target > Debug Active Project** from the CCSv4/CCsv5 menus. If this is the first time you are debugging a project for your target, you may need to set up a CCS Target Configuration. See the CCSv4/v5 help for details.
3. In the Debug perspective, open the Runtime Object Viewer (ROV) tool by choosing **Tools > ROV**. Also open the Raw Logs view by choosing **Tools > RTA > Raw Logs**. These tools allow you to see the activity of RTSC and SYS/BIOS modules.
4. Set some breakpoints in the log.c source file. (You can do this by right-clicking on a line and choosing **New Breakpoint > Breakpoint**.) For example, set a breakpoint on the last line of each function in log.c.
5. Run the application.
6. In the Raw Logs window, you can see the informational, warning, and error messages sent by the calls to Log module APIs in log.c. The messages that begin with “LM” are diagnostics provided by XDCtools. Messages that begin with “WARNING” come from calls to Log_warning2. Messages that begin with “ERROR” come from calls to Log_error2. Messages that begin with “../log.c” come from calls to Log_info0 and Log_info2 (depending on the number of arguments).



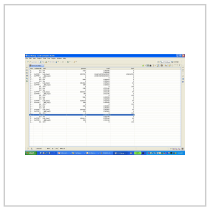
SYSBIOS Diagnostics



SYSBIOS_Rawlog



SYSBIOS_Execution_Graph



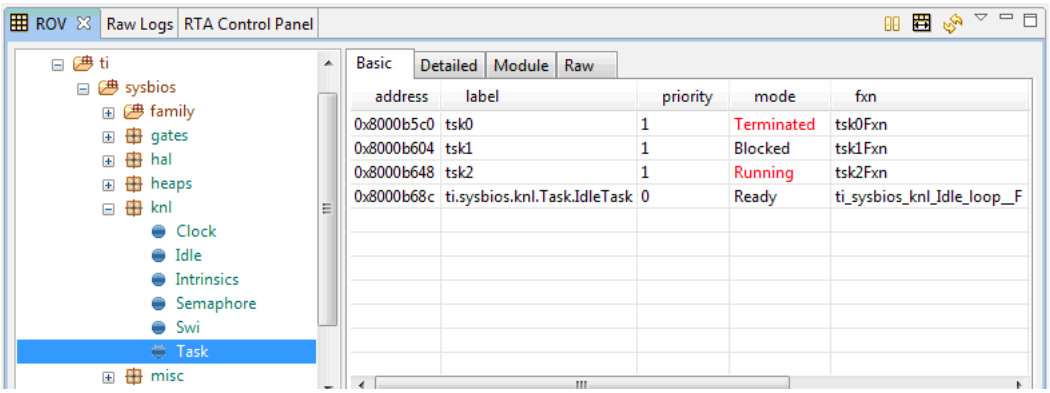
SYSBIOS_CPU_Load

7. For advanced debugging options we recommend following the instructions on the BIOS_6_Real-Time_Analysis_(RTA)_in_CCSv4 (http://processors.wiki.ti.com/index.php/BIOS_6_Real-Time_Analysis_(RTA)_in_CCSv4) wiki

RTOS Object View(ROV)

1. Load your application for debugging. Select the device you want to debug before opening ROV.

2. In the ROV window, expand the tree to see the ti.sysbios.knl.Task module. The right pane shows a list of the Task threads in the application. As you advance from breakpoint to breakpoint, you see the run mode of the threads change.



For more details on ROV tools, refer to the Runtime Object Viewer(ROV) (http://rtsc.eclipse.org/docs-tip/Runtime_Object_Viewer) article on RTSC website.

Exception handling

SYS/BIOS provides several target unique exception handlers:

- ti.sysbios.family.arm.exc.Exception - used by all Arm9 targets
- ti.sysbios.family.c64p.Exception - used by all C6x targets

By default, beginning with the 6.32 SYS/BIOS releases, all of these exception handlers print a complete register context dump to the CCS console when an exception is detected. If you copy the exception dump's values for the PC, SP, (and LR for ARM targets) into the corresponding registers in the CCS register view, CCS will usually provide a very useful call stack back trace in the debug window.

Linux Debugging using CCSv4/v5

This feature is only supported in CCSv5.

- Instructions to do a Linux aware application debugging in CCSv4 are described here (http://processors.wiki.ti.com/index.php/Linux_Aware_Debug_(CCSv4.x)).
- Instructions to setup and debug a linux application have been explained in detail on Linux Debug in CCSv5 (http://processors.wiki.ti.com/index.php/Linux_Debug_in_CCSv5) wiki.

Other useful debugging resources

Debugging GEL file for boot/application crash related issues

The Debug GEL file can be used to find Boot modes, Device and ROM Ids, Current state and other useful information while debugging. Instruction to use this GEL file in CCS have been archived on the OMAPL Debug GEL file (http://processors.wiki.ti.com/index.php/OMAP-L1x_Debug_Gel_Files) wiki.

Exception handling in non-OS environment

Resource conflicts, illegal opcodes, etc generate internal exceptions on the C674x. In order to catch these exceptions you can use the following sample code. The sample code contains includes main() function just sets up the exception logging and interrupt vector tables and then intentionally forces an internal exception by running bad opcodes.

- Sample code: File:Exception handling.zip

Useful Training links/Presentations

- ETB (http://processors.wiki.ti.com/index.php/Embedded_Trace_Buffer)
- Debugging with Trace (http://processors.wiki.ti.com/index.php/Debugging_With_Trace)
- Unified Break Point manager (http://processors.wiki.ti.com/index.php/Unified_Breakpoint_Manager)
- File:SYSBIOS 05 RTA.zip
- Trace_CCSv4 (Not OMAPL specific) (http://software-dl.ti.com/dsps/dsps_public_sw/sdo_ccstudio/CCSv4/Demos/TraceCommon_ETB.htm)]

{{ 1. switchcategory:MultiCore= • For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum • For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **Debug and Trace Capabilities on**	Keystone= • For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum • For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum	C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article* ***Debug and Trace Capabilities on OMAPL1X/AM1x/DA8x devices*** *here.*	DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article* ***Debug and Trace Capabilities on OMAPL1X/AM1x/DA8x devices*** *here.*	MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article* ***Debug and Trace Capabilities on OMAPL1X/AM1x/DA8x devices*** *here.*	OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***Debug and Trace Capabilities on OMAPL1X/AM1x/DA8x devices*** *here.*	Ol *su* *pl* *qu* *O* *po* ***ab*** ***ar*** ***o*** ***O*** ***de***

OMAPL1X/AM1x/DA8x devices here.	Please post only comments related to the article **Debug and Trace Capabilities on OMAPL1X/AM1x/DA8x devices** here.

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity