Debugging Boot Issues

From Texas Instruments Wiki
Jump to: navigation, search

Contents

[hide]

Intro[edit]

This article applies to processors that have only internal RAM (no integrated flash) that must be booted from external non-volatile memory.

Often times everything will work great when running code from the emulator, but when you get to that last crucial step of having the device boot and run by itself things mysteriously fail.

In these scenarios the key is that we want to use CCS to debug the issue, but unlike earlier during the development phase we want to simply **observe** what is happening. In

other words, we no longer want CCS actively configuring registers, loading code, etc. We simply want the board to boot from flash while we observe what it is doing with CCS. In order to do this properly there are a few key things that need to be done slightly different than when you were developing your code.

Key Steps[edit]

Be careful with gel files[edit]

Many gel files are intrusive on the target. For example, they might configure the PLL and/or External Memory Interface (EMIF). If you're having issues with getting your device to work properly when it boots from flash, you should double-check that you're not reliant on the gel file to setup something critical like the PLL or EMIF. Also, you don't want the gel file to modify the way the processor runs when trying to debug issues. You want the processor to boot exactly as it always does when debugging from flash.

More info on gel files can be found in these articles:
GEL
FAQ - CCSv4#GEL

"Load Symbols" instead of "Load Program"[edit]

When debugging an application from flash, you want to let the application boot in its normal manner. If you select "load program" in CCS then you are overwriting the application that loaded from flash and not debugging the code as it runs normally. You should instead do "load symbols" in CCS and then select your .out file. This will allow you to debug your code using variable/function names without overwriting the code that boots from the flash.

- CCS 3.3: Go to File -> Load Symbols -> Load Symbols Only
- CCS 4.x: Right-click on the project and select Debug Options. On the "Debugger" tab choose "Load Symbols" instead of "Load Program"
- CCS 5.x: In the "Debug View" tab choose "Run"--> "Load" --> "Load Symbols"

Use Hardware Breakpoints[edit]

A software breakpoint works by using a special opcode, or modifying an opcode. A benefit to software breakpoints is that there's no limit to the number you can use. The drawback is that when the code gets reloaded (e.g. when the bootloader codes runs after a reset) the software breakpoints get overwritten.

If you're allowing the device to boot from flash, you'll need to have your FIRST breakpoint be a HARDWARE breakpoint in order to force the target to halt. Just right-click and select "Toggle Hardware Breakpoint" in CCS to set a hardware breakpoint.

You can still use software breakpoints once you've hit that first hardware breakpoint. After a reset, however, you need to "refresh" all the software breakpoints.

- First, open the Breakpoint Manager
  - CCS 3.3: Go to Debug -> Breakpoints
  - CCS 4.x: Go to View -> Breakpoints
- In the breakpoint manager you can click "disable all" and "enable all" such that CCS will set them again.

CCS Crashing when Connecting[edit]

There's an issue related to RTDX and DSP/BIOS where you might see an emulator error when trying to connect to your device that's booting from flash. This issue is corrected for 64x+ devices starting with DSP/BIOS 5.31. For all other devices and for older versions of DSP/BIOS you can only work around this issue by disabling RTDX in your BIOS configuration.

To disable RTDX make sure the following are properly setup in the BIOS configuration:

- System --> Global Settings: Uncheck "Enable RTA" and "Enable TRC Trace Event Classes"
- Input/Output --> HST: host link type set to "NONE"
- Input/Output --> RTDX: Uncheck "Enable Real-Time Data Exchange"

Debugging problems with the bootloader [edit]

Many devices require you to write your own bootloader (e.g. c6713, c6416, DM642, etc.). An excellent reference for creating this bootloader is spra999, and here is the accompanying example code. In order to debug a bootloader that you've created you may need to put an infinite spin loop at the beginnning (temporarily). You could implement this loop as follows:

```
start_of_bootloader:
          ZERO B1
spin:

    [!B1] B spin

          NOP 5
```

Once you've attached CCS then you can open a register window and set B1 to a nonzero value in order to exit the loop.

{{

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 **Debugging Boot Issues** here.

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

Please post only comments related to the article **Debugging Boot Issues** 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

Categories:

- Emulation
- DSPBIOS
- Code Composer Studio v3
- Code Composer Studio v4
- Debug
- DaVinci Debugging

Navigation menu

Personal tools

- Log in
- Request account

Namespaces

- Page
- Discussion

Variants

Views

- Read
- View source
- View history

More

Search

Go

Navigation

- Main Page
- All pages

- All categories
- Recent changes
- Random page
- Help

Toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link
- Page information

- This page was last edited on 18 July 2012, at 16:28.
- Content is available under Creative Commons Attribution-ShareAlike unless otherwise noted.

- Privacy policy
- About Texas Instruments Wiki
- Disclaimers
- Terms of Use
- 
-