Debugging UBoot with the BDI2000/BDI3000

BDI2000/BDI3000 CONFIGURATION

Make a config file that does the minimum register initializations to get Uboot onto your board. If you already have Uboot in flash than you can use a blank [INIT] section that just holds the system at the reset vector. If you are debugging Uboot from flash make sure to define BREAKMODE HARD in the [TARGET] section of the BDI. In some targets you may need some minimal initialization to hold the target at reset. Disable the watchdog timer if it is present. In some targets you may need to set up and initial TLB mapping for your boot space before you can access memory.

UBOOT CONFIGURATION

Compile Uboot with the -g compiler flag option to create an image with the debug symbols in it. Then create a stripped version to load onto your target using the "objcopy -g" command. It is recommended that you also remove any optimizations (-O, -O2) from the compiler when you are debugging as they tend to interfere with GDB's single step capabilities. The best way to make all these changes is to go into the config.mk and edit the OPTFLAGS to be:

OPTFLAGS= -Os -fno-schedule-insns -fno-schedule-insns2 #-fomit-frame-pointer -g

You will also need to know what ram address Uboot relocates itself to. This will be used later in debugging during step 11. The ram relocation address for the powerpc comes from “lib_ppc\board.c\board_init_f()”, the statement(s):

len = (ulong)&_end - CFG_MONITOR_BASE
```
addr = CFG_SDRAM_BASE + get_effective_memsize()
addr -=len
```

Variable' end' is initialized to '0x0. CFG_SDRAM_BASE can be found in /include/configs/yourboard.h and is normally set to 0x0. CFG_MONITOR_BASE is the TEXT_BASE from your /config/yourboard/config.mk. get_effective_memsize() will return the total size of your SDRAM. With the above assumptions the ram relocation address becomes the hex arthmetic:

addr = Size of Memory in Hex + TEXT_BASE

HOST CONFIGURATION

Make sure you have a version of GDB compiled that will talk to your target system and works on your host. See http://www.ultsol.com/pdfs/Tool_Talk_03-001-Using-GDB-for-PowerPc.pdf for details.

PROCEDURE FOR UBOOT DEBUGGING

1.) Connect the BDI2000/BDI3000 to the Target
2.) Power up the BDI2000/BDI3000 first and start a telnet session. Let it load the config file when it says waiting for target VCC
3.) Power up the Target
4.) At this point the default behavior is Reset Halt, which means the target will not be running. Put 'i' at the BDI prompt to make sure the target is not running and sitting at your boot vector then
5.) Type in "load" at the BDI prompt and load Uboot (stripped version) onto your target system or ignore this step if Uboot is already loaded into flash.
6.) Use a few "ti" commands from the BDI prompt to confirm that you can step through the target, the BDI prompt will tell you that it is stepping,
    BDI> ti
    BDI> ti
7.) On the host platform, start GDB with the larger Uboot image located in the root of the Uboot directory tree, this has symbols (the "–g" option added to the make file);
    [host] gdb /path_to/Uboot
8.) Connect to the target using:
    (gdb) target remote 192.168.123.105:2001
    Where 192.168.123.105 is the IP you configured the BDI with
9.) Perform a few steps using "stepi" in the GDB prompt, you should see a step occur in the BDI Prompt;
    (gdb) stepi
    (gdb) stepi
    Also perform a "list" command in the GDB prompt and make sure GDB knows where it is,
    (gdb) list
10.) Finally, set your own break points in GDB and see if it the BDI2000/BDI3000 hits the breakpoints. If the breakpoint does not hit go to step 11.
    (gdb) b symbol_name
    (gdb) c
11.) If Uboot is loaded from Flash then at some point it relocates itself to RAM but GDB is not aware of this. You need to manually remove the current symbols from GDB and add the new symbols with the relocated RAM address (this address is displayed by Uboot during boot up):
    (gdb) symbol-file
    (gdb) add-symbol-file /path_to/Uboot 0xfd0000
    Where 0xfd0000 is the relcoated .text address
12.) Once GDB is aware of the relocated RAM address you can set breakpoints;
    (gdb) b symbol_name
    (gdb) c

NOTE:
To be able to view global and local variables you will need to change the command in step 11 to:

(gdb) add-symbol-file /path_to/Uboot 0xfd0000
-s .rodata 0xcf030354\
-s .data 0xcf030488\
-s .sdata 0xcf030488\
-s .bss 0xcf030519\
-s .sbss 0xcf03051c

You can find all the relevant address from the .map file created when uboot is compiled. However once Uboot relocates to ram these addresses are invalid. To calculate the correct addresses you need to find the offset between the flash address and the ram address then add it into the .rodata, .data, .sdata, .bss and .sbss addresses.

Authored by: Fahd Abidi
Field Application Engineer with Ultimate Solutions, Inc.

10 Clever Lane
Tewksbury, MA 01876-1580 USA
Ph: 866.455.3383 Fx: 978.926.3091
Email: info@ultsol.com
Web: www.ultsol.com