Kernel Course: Lecture 19

Hardware Tools in Embedded Kernel Development

Sam Protsenko

April 15, 2019

GlobalLogic

Agenda

Hardware Tools Overview

Where it can be useful?

- Debugging hardware issues (obviously)
- Board bringup (measuring clocks, voltages, etc)
- Fixing adapter drivers
- Performance optimizations
- Power management optimizations
- Investigating low-level components
- Debugging subtle problems

Multimeter

Multimeter Overview

- Measure voltages:
  - Can be used for power issues diagnostics
  - Helpful during board bring-up
- Measure consumed current:
  - Useful for power management improvements

Current Measurement

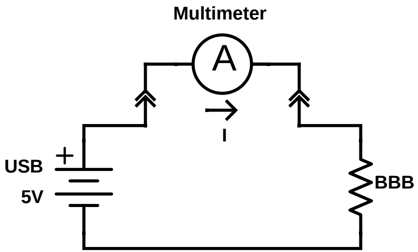


Figure 1: Ammeter Connection

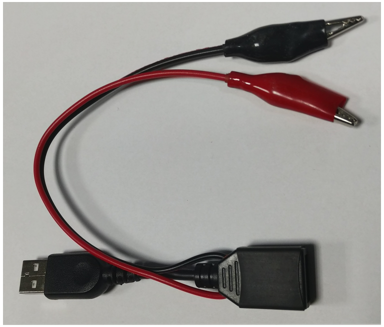


Figure 2: Measurement Cable

Power can be calculated:

$$P = V \cdot I$$

Multimeter Demo: Suspend/Resume Current

Scope

Scope Overview

- Software vs hardware
- Parameters: frequency, data rate; + probes (capacity)
- Measuring periodic signals
- Trigger feature
- Divisors (/1, /10)
- 2 channels (common ground!)
- Scale: time, voltage
- “Auto” button
- “Run/stop” button
- Measure mode

How it can be useful for kernel development?

Demo 1

Scope Demo #1: Measuring Delay

Demo 2

Scope Demo #2: Investigating Kernel Sleeps

Logic Analyzer

Logic Analyzer Overview

- Catches digital levels
- Stores collected data to internal (hardware) buffer
- App can parse protocols (I2C, UART, etc)
- Parameters: max data rate (freq), ports count, protocols support
- Useful for debugging (sw, hw) and reverse engineering ("sniffing")
- Some specialized LAs exist (like USB ones)

How it can be useful for kernel development?

Demo 1

LA Demo #1: I2C (SDA, SCL)

Demo 2

LA Demo #2: UART (Tx, Rx)

Take Five

JTAG

JTAG Overview

- OpenOCD: Flyswatter (FT2232H based, MPSSE)
- Proprietary JTAG: XDS100v2
- Allows one to debug program running on board just like user-space application with GDB:
  - Breakpoints
  - Read/write registers
  - Stop/run CPU
  - Read/write memory
  - Load images
  - Examine stack
  - Can debug U-Boot, kernel, user-space apps/libs, even ROM-code...

OpenOCD

OpenOCD Overview

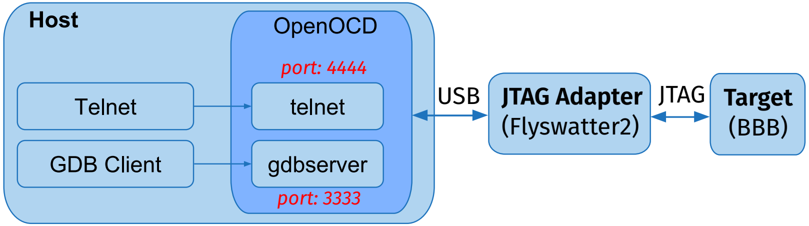


Figure 3: Debugging BBB with OpenOCD

OpenOCD creates servers for:

- Telnet (port 4444 on localhost)
- GDB (port 3333 on localhost)

Once OpenOCD is running, you can run:

```
$ telnet localhost 4444
  or
$ arm-eabi-gdb vmlinux
```

Flyswatter2 Overview

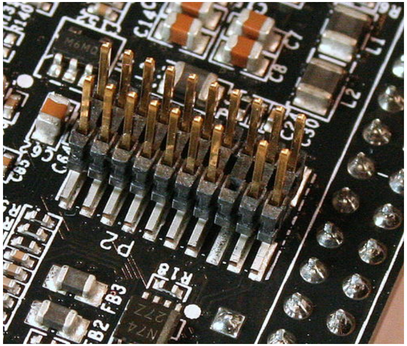


Figure 4: BBB JTAG Connector

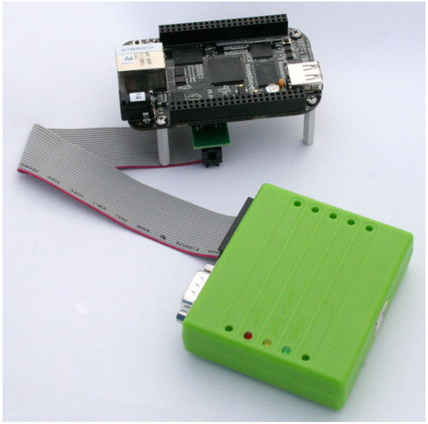


Figure 5: BBB and Flyswatter2

Kernel Preparations

- Enable `CONFIG_DEBUG_INFO` (the same as `-g`)
- Disable `CONFIG_WATCHDOG` (just in case)
- Make sure JTAG clock (DEBUGSS on BBB) is alive in kernel (see next slide)
- Rebuild, add `zImage` to rootfs, flash to BBB

Kernel Preparations (cont'd)

arch/arm/mach-omap2/omap_hwmod_33xx_data.c:

```
static struct omap_hwmod am33xx_debugss_hwmod = {
        .name           = "debugss",
        .class          = &am33xx_debugss_hwmod_class,
        .clkdm_name     = "l3_aon_clkdm",
+       .flags          = (HWMOD_INIT_NO_IDLE | HWMOD_INIT_NO_RESET),
        .main_clk       = "trace_clk_div_ck",
        .prcm           = {
```

Without this patch you'll see `STICKY BIT` errors (in OpenOCD terminal) when kernel starts running.

OpenOCD Preparations

Install OpenOCD (my version is 0.10.0):

```
$ sudo apt update
$ sudo apt install openocd
```

- Figure out correct OpenOCD interface and target config files:
    - `interface/ftdi/flyswatter2.cfg`
    - `target/am335x.cfg`
- ...but those files from upstream OpenOCD won’t work
- Use Flyswatter2 config from TinCanTools site (see next page)

OpenOCD Working Script for Flyswatter2 (1/5)

Listing 1: ti_beaglebone_with_fs2_mod.cfg

```
# AM335x Beaglebone, for use with the TinCanTools Flyswatter2
#  http://beagleboard.org/bone
#  http://www.tincantools.com

# The JTAG interface is built directly on the board.
interface ftdi
ftdi_device_desc "Flyswatter2"
ftdi_vid_pid 0x0403 0x6010

ftdi_layout_init 0x0538 0x057b
ftdi_layout_signal LED -ndata 0x0400
ftdi_layout_signal nTRST -data 0x0010
ftdi_layout_signal nSRST -data 0x0020 -noe 0x0100

adapter_khz 16000


if { [info exists CHIPNAME] } {
   set _CHIPNAME $CHIPNAME
} else {
```

OpenOCD Working Script for Flyswatter2 (2/5)

```
    set _CHIPNAME am335x
}

# This chip contains an IcePick-D JTAG router.  The IcePick-C configuration is almost
# compatible, but it doesn't work.  For now, we will just embed the IcePick-D
# routines here.
proc icepick_d_tapenable {jrc port} {
        # select router
        irscan $jrc 7 -endstate IRPAUSE
        drscan $jrc 8 0x89 -endstate DRPAUSE

        # set ip control
        irscan $jrc 2 -endstate IRPAUSE
        drscan $jrc 32 [expr 0xa0002108 + ($port << 24)] -endstate DRPAUSE

        # for icepick_D
        irscan $jrc 2 -endstate IRPAUSE
        drscan $jrc 32 0xe0002008 -endstate DRPAUSE

        irscan $jrc 0x3F -endstate RUN/IDLE
        runtest 10
}

```

OpenOCD Working Script for Flyswatter2 (3/5)

```
#
# M3 DAP
#
if { [info exists M3_DAP_TAPID] } {
        set _M3_DAP_TAPID $M3_DAP_TAPID
} else {
        set _M3_DAP_TAPID 0x4b6b902f
}
jtag newtap $_CHIPNAME m3_dap -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id $_M3_DAP_TAPID -disable
jtag configure $_CHIPNAME.m3_dap -event tap-enable "icepick_d_tapenable $_CHIPNAME.jrc 11"

#
# Main DAP
#
if { [info exists DAP_TAPID] } {
   set _DAP_TAPID $DAP_TAPID
} else {
   set _DAP_TAPID 0x4b6b902f
}
jtag newtap $_CHIPNAME dap -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id $_DAP_TAPID -disable
jtag configure $_CHIPNAME.dap -event tap-enable "icepick_d_tapenable $_CHIPNAME.jrc 12"

#
```

OpenOCD Working Script for Flyswatter2 (4/5)

```
# ICEpick-D (JTAG route controller)
#
if { [info exists JRC_TAPID] } {
   set _JRC_TAPID $JRC_TAPID
} else {
   set _JRC_TAPID 0x0b94402f
}
jtag newtap $_CHIPNAME jrc -irlen 6 -ircapture 0x1 -irmask 0x3f -expected-id $_JRC_TAPID -ignore-version
jtag configure $_CHIPNAME.jrc -event setup "jtag tapenable $_CHIPNAME.dap"
# some TCK tycles are required to activate the DEBUG power domain
jtag configure $_CHIPNAME.jrc -event post-reset "runtest 100"

#
# Cortex A8 target
#
set _TARGETNAME $_CHIPNAME.cpu
target create $_TARGETNAME cortex_a8 -chain-position $_CHIPNAME.dap -dbgbase 0x80001000

# SRAM: 64K at 0x4030.0000; use the first 16K
$_TARGETNAME configure -work-area-phys 0x40300000 -work-area-size 0x4000

$_TARGETNAME configure -event gdb-attach {
   cortex_a dbginit
```

OpenOCD Working Script for Flyswatter2 (5/5)

```
    halt
}


reset_config trst_and_srst
```

Run OpenOCD

1. Connect Flyswatter2 to BBB using adapter kit
2. Connect Flyswatter2 to PC via USB
3. Connect BBB to PC via USB (power);
   “TARGET RESET” LED on Flyswatter is glowing, CPU is in reset state
4. Run OpenOCD command:

```
$ openocd -f ./ti_beaglebone_with_fs2_mod.cfg -c "init" -c "reset init"
```

   We will use this terminal to track OpenOCD log.

Correct OpenOCD Command Output

```
adapter speed: 16000 kHz
Info : auto-selecting first available session transport "jtag". To override use 'transport select <transport↩
      >'.
Warn : target name is deprecated use: 'cortex_a'
trst_and_srst separate srst_gates_jtag trst_push_pull srst_open_drain connect_deassert_srst
Info : ftdi: if you experience problems at higher adapter clocks, try the command "ftdi_tdo_sample_edge ↩
      falling"
Info : clock speed 16000 kHz

Info : JTAG tap: am335x.jrc tap/device found: 0x2b94402f (mfg: 0x017 (Texas Instruments), part: 0xb944, ver: 0x2)
Info : JTAG tap: am335x.dap enabled
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Error: target->coreid 0 powered down!

Info : JTAG tap: am335x.jrc tap/device found: 0x2b94402f (mfg: 0x017 (Texas Instruments), part: 0xb944, ver: 0x2)
Info : JTAG tap: am335x.dap enabled
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : am335x.cpu: hardware has 6 breakpoints, 2 watchpoints
```

OpenOCD over GDB (page 1)

Let’s start debug session in GDB:

```
    $ arm-eabi-gdb vmlinux

    (gdb) target remote localhost:3333
    (gdb) monitor cortex_a dacrfixup on
    (gdb) continue
```

What we did here:

1. Connect to OpenOCD’s GDB server, using Linux kernel symbols
2. Using OpenOCD `cortex_a` command, do a workaround for software breakpoints (Linux kernel maps `.text` read-only, and the debugger cannot write a breakpoint due to that)
3. Continue CPU execution (as it was in reset state initially);
   we use GDB commands for debugging (not `monitor resume`, etc)

OpenOCD over GDB (page 2)

Set a breakpoint and trigger it:

```
^C
(gdb) break do_sys_open
(gdb) continue

/ # cat /proc/cmdline
```

1. Stop CPU execution by pressing `Ctrl-C`
2. Set breakpoint for `do_sys_open()` kernel function (it's a handler for `open()` syscall. Another interesting functions would be `load_module()`, `start_kernel()`, etc
3. Continue CPU execution
4. Print `/proc/cmdline` file, so that `open()` is triggered
5. Now we are in breakpoint, CPU is halted

OpenOCD over GDB (page 3)

Investigate code in breakpoint:

```
(gdb) monitor cortex_a maskisr on
(gdb) continue
(gdb) ...
(gdb) bt, list, info registers, disas, step, print, ...
(gdb) monitor cortex_a maskisr off
(gdb) continue
```

1. Don’t process interrupts when stepping (otherwise we won’t be able to perform `continue`)
2. Wait for `open("/proc/cmdline")`
3. Investigate caught code
4. Once we are done, enable interrupts processing and continue normal execution

OpenOCD Telnet Commands

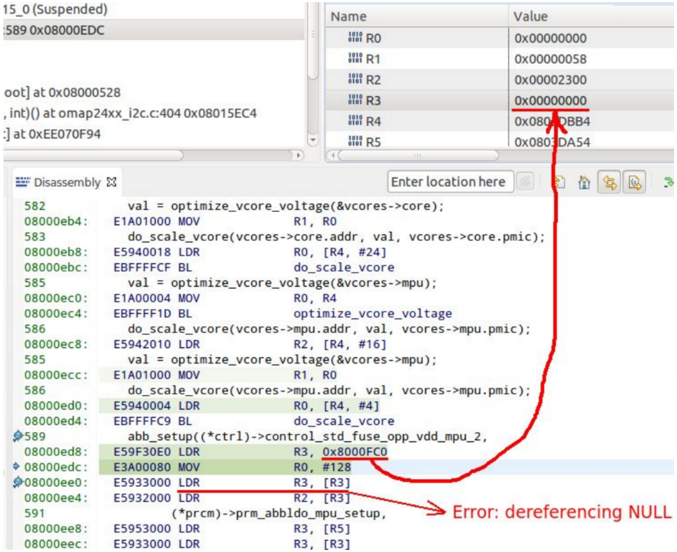
In Telnet, you can use regular OpenOCD monitor commands:

- help
- reset
- halt
- resume
- reg
- bp <address> <len> [hw]
- step [address]
- mdw, mww

See OpenOCD User’s Guide for details.

XDS100v2

Proprietary JTAG: XDS100v2 via CodeComposerStudio



```
15_0 (Suspended)
:589 0x08000EDC

oot] at 0x08000528
, int)() at omap24xx_i2c.c:404 0x08015EC4
:] at 0xEE070F94
```

Name	Value
R0	0x00000000
R1	0x00000058
R2	0x00002300
R3	0x00000000
R4	0x080?DBB4
R5	0x0803DA54

```
582           val = optimize_vcore_voltage(&vcores->core);
08000eb4:   E1A01000 MOV           R1, R0
583           do_scale_vcore(vcores->core.addr, val, vcores->core.pmic);
08000eb8:   E5940018 LDR           R0, [R4, #24]
08000ebc:   EBFFFFCF BL            do_scale_vcore
585           val = optimize_vcore_voltage(&vcores->mpu);
08000ec0:   E1A00004 MOV           R0, R4
08000ec4:   EBFFFF1D BL            optimize_vcore_voltage
586           do_scale_vcore(vcores->mpu.addr, val, vcores->mpu.pmic);
08000ec8:   E5942010 LDR           R2, [R4, #16]
585           val = optimize_vcore_voltage(&vcores->mpu);
08000ecc:   E1A01000 MOV           R1, R0
586           do_scale_vcore(vcores->mpu.addr, val, vcores->mpu.pmic);
08000ed0:   E5940004 LDR           R0, [R4, #4]
08000ed4:   EBFFFFC9 BL            do_scale_vcore
589           abb_setup((*ctrl)->control_std_fuse_opp_vdd_mpu_2,
08000ed8:   E59F30E0 LDR           R3, 0x8000FC0
08000edc:   E3A00080 MOV           R0, #128
08000ee0:   E5933000 LDR           R3, [R3]
08000ee4:   E5932000 LDR           R2, [R3]
591                 (*prcm)->prm_abbldo_mpu_setup,
08000ee8:   E5953000 LDR           R3, [R5]
08000eec:   E5933000 LDR           R3, [R3]
```

Demo

JTAG Demo #1: Flyswatter2

Demo

JTAG Demo #2: XDS560v2

JTAG References

- https://www.tincantools.com/flyswatter2-beaglebone-black-how-to/
- https://e2e.ti.com/support/embedded/linux/f/354/t/363421
- https://www.tincantools.com/beaglebone-black-eclipse-gdb/
- https://devel.rtems.org/wiki/Debugging/OpenOCD/BeagleBoneBlack
- https://elinux.org/Debugging_The_Linux_Kernel_Using_Gdb
- https://github.com/n-aizu/freertos-multicore/wiki/Debugging-with-openocd
- http://openocd.org/doc/html/GDB-and-OpenOCD.html
- http://openocd.org/doc/html/General-Commands.html
- https://habr.com/post/206036/

Thank you!