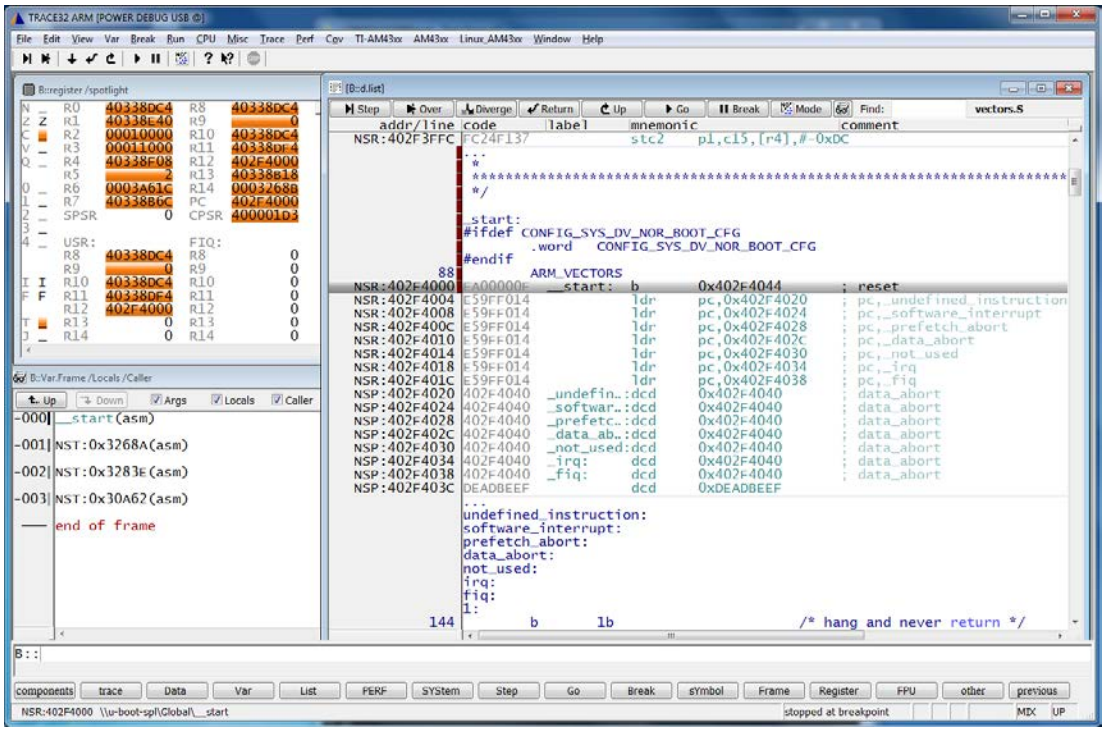
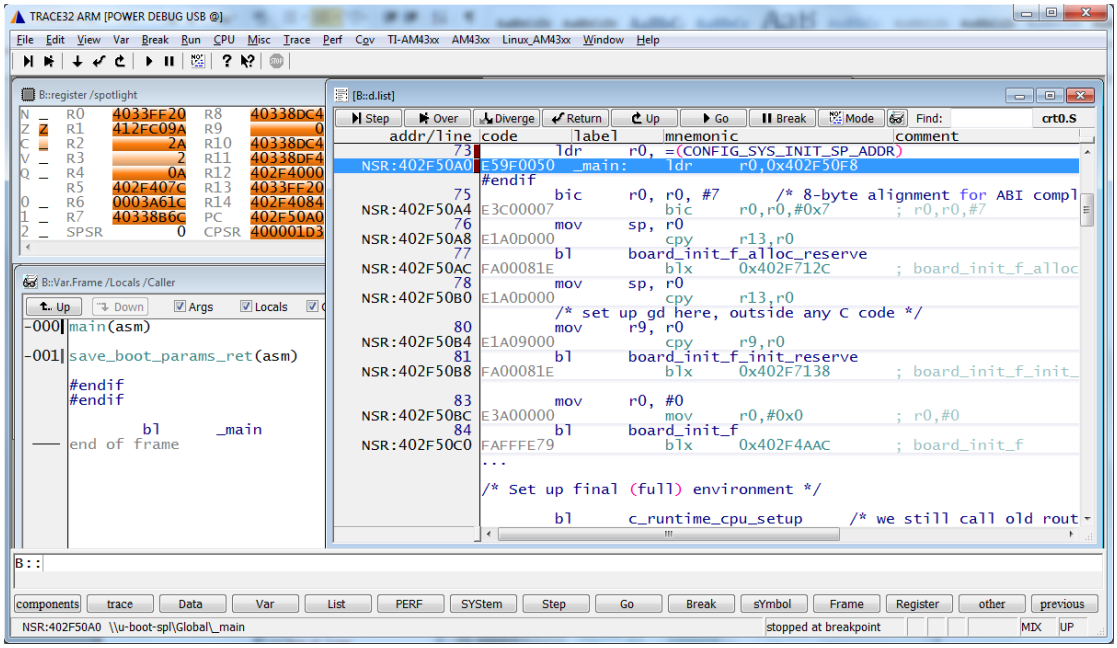
SPL Boot Flow (AM437x GP is used as an example)

1. At the SPL code entry @0x402F4000



2. At the entry of ENTRY(_main) in crt0.S (arch/arm/lib/crt0.S)



3. arch/arm/lib/crt0.S (a good summary of SPL boot flow)

```
/*
 * This file handles the target-independent stages of the U-Boot
 * start-up where a C runtime environment is needed. Its entry point
 * is _main and is branched into from the target's start.S file.
 *
 * _main execution sequence is:
 *
 * 1. Set up initial environment for calling board_init_f().
 *    This environment only provides a stack and a place to store
 *    the GD ('global data') structure, both located in some readily
 *    available RAM (SRAM, locked cache...). In this context, VARIABLE
 *    global data, initialized or not (BSS), are UNAVAILABLE; only
 *    CONSTANT initialized data are available. GD should be zeroed
 *    before board_init_f() is called.
 *
 * 2. Call board_init_f(). This function prepares the hardware for
 *    execution from system RAM (DRAM, DDR...) As system RAM may not
 *    be available yet, , board_init_f() must use the current GD to
 *    store any data which must be passed on to later stages. These
 *    data include the relocation destination, the future stack, and
 *    the future GD location.
 *
 * 3. Set up intermediate environment where the stack and GD are the
 *    ones allocated by board_init_f() in system RAM, but BSS and
 *    initialized non-const data are still not available.
 *
 * 4a.For U-Boot proper (not SPL), call relocate_code(). This function
 *    relocates U-Boot from its current location into the relocation
 *    destination computed by board_init_f().
 *
 * 4b.For SPL, board_init_f() just returns (to crt0). There is no
 *    code relocation in SPL.
 *
 * 5. Set up final environment for calling board_init_r(). This
 *    environment has BSS (initialized to 0), initialized non-const
 *    data (initialized to their intended value), and stack in system
 *    RAM (for SPL moving the stack and GD into RAM is optional - see
 *    CONFIG_SPL_STACK_R). GD has retained values set by board_init_f().
 *
 * 6. For U-Boot proper (not SPL), some CPUs have some work left to do
 *    at this point regarding memory, so call c_runtime_cpu_setup.
 *
 * 7. Branch to board_init_r().
 *
 * For more information see 'Board Initialisation Flow in README.
 */
```

4. Boot flow details

a) arch/arm/lib/crt0.S

main (asm)
{
…
board_init_f() -> sdram_init() -> config_ddr()
board_init_r() -> spl_board_init() ….. load/run u-boot @0x80800000 in DDR
}

b) core board file (/arch/arm/mach-omap2/am33xx/board.c)

board_init_f()
{
early_system_init();
board_early_init_f();
sdram_init();
gd->ram_size = get_ram_size();
}

c) AM43xx board file (/board/ti/am43xx/board.c)

sdram_init()

d) core DDR files

/arch/arm/mach-omap2/am33xx/ddr.c
/arch/arm/mach-omap2/am33xx/emif4.c
config_ddr()

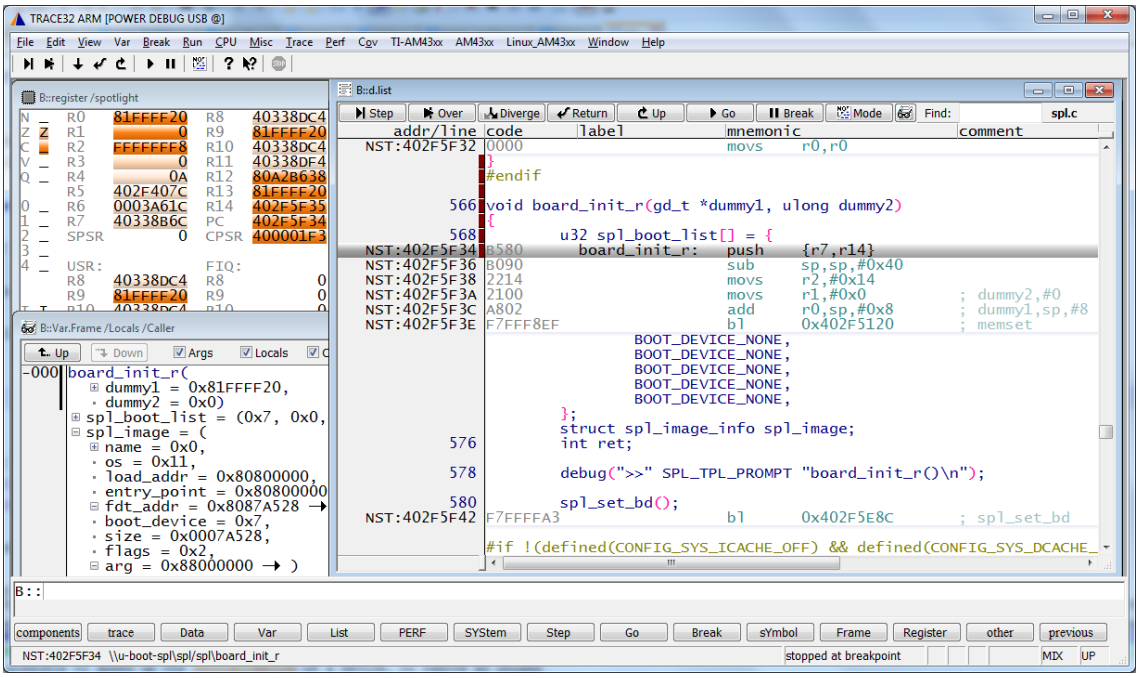
e) common SPL file (common/spl/spl.c)

board_init_r()

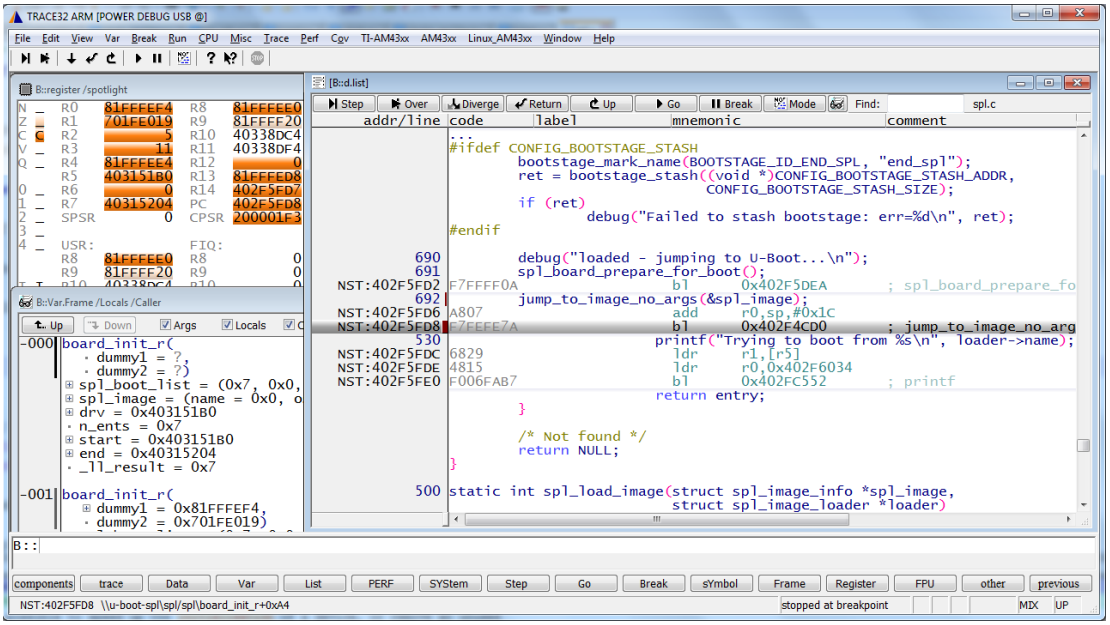
f) (arch/arm/mach-omap2/boot-common.c)

spl_board_init()

5. At the entry of board_init_r()



6. Right before switching to u-boot, still in board_init_r()



7. At the entry of u-boot @0x8080000 in DDR