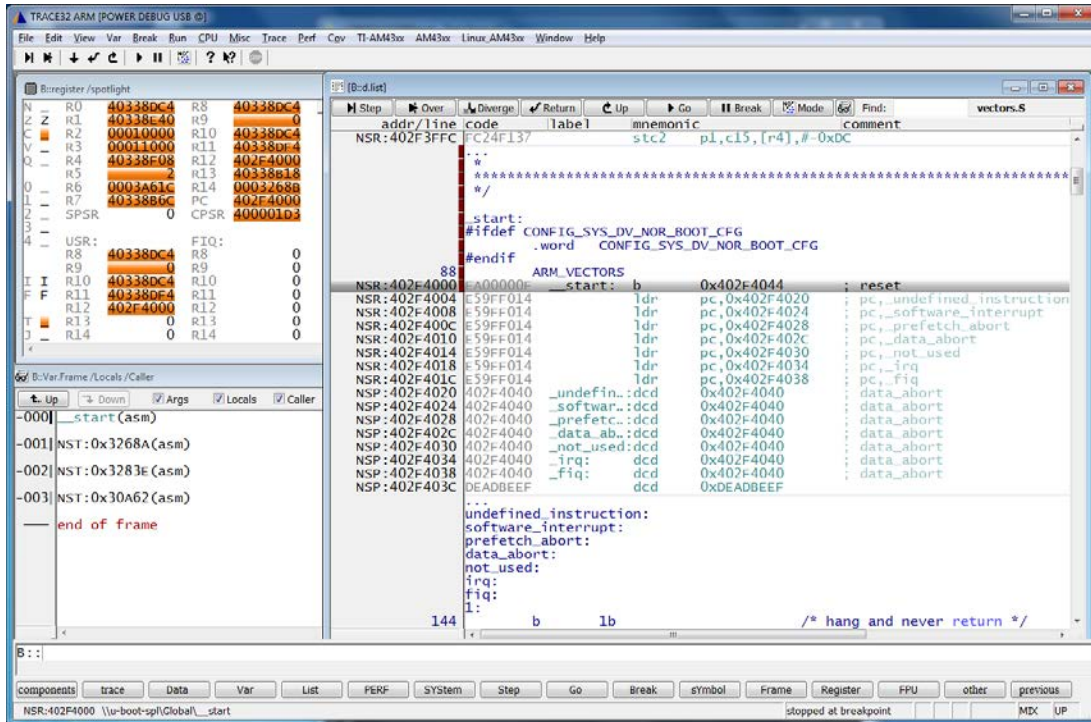
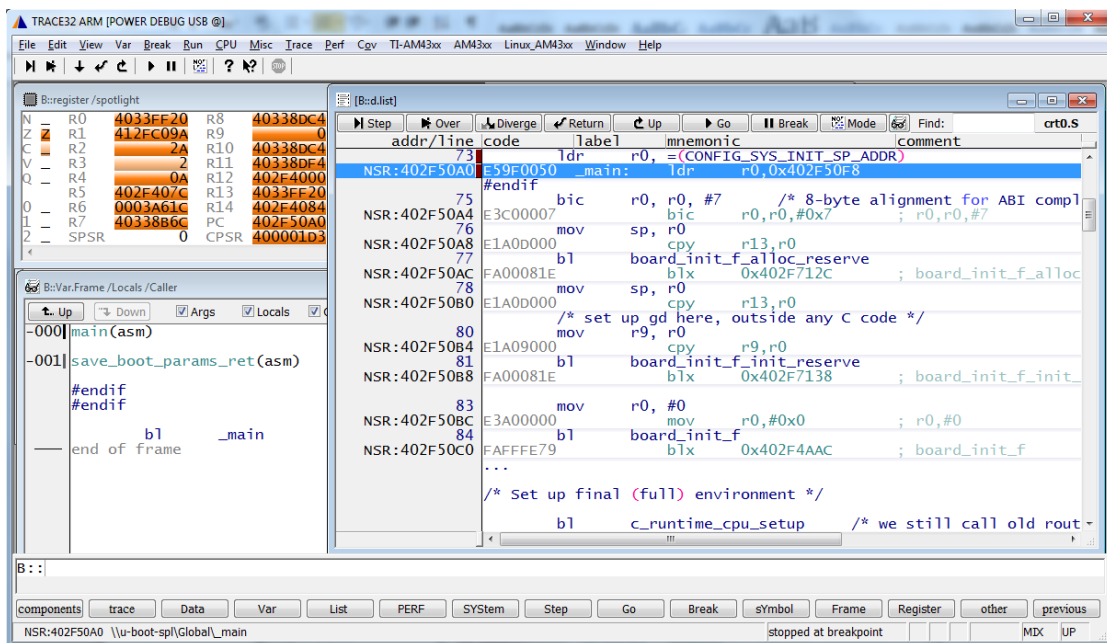


SPL Boot Flow (AM437x GP is used as an example)

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()

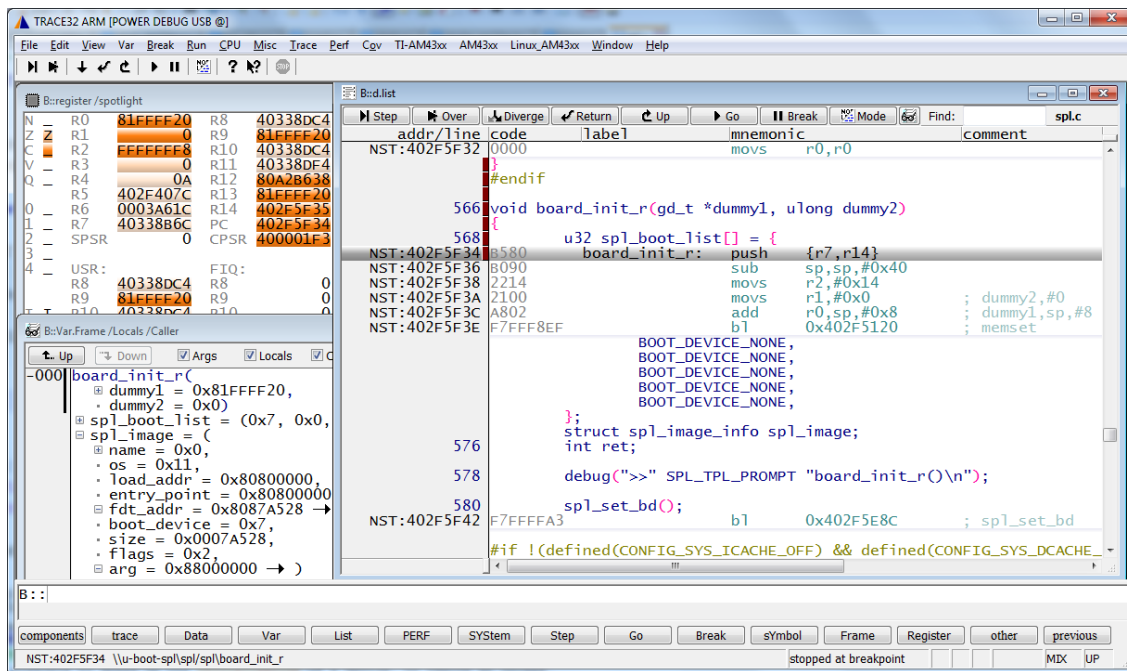
e) common SPL file (common/spl/spl.c)

board_init_r()

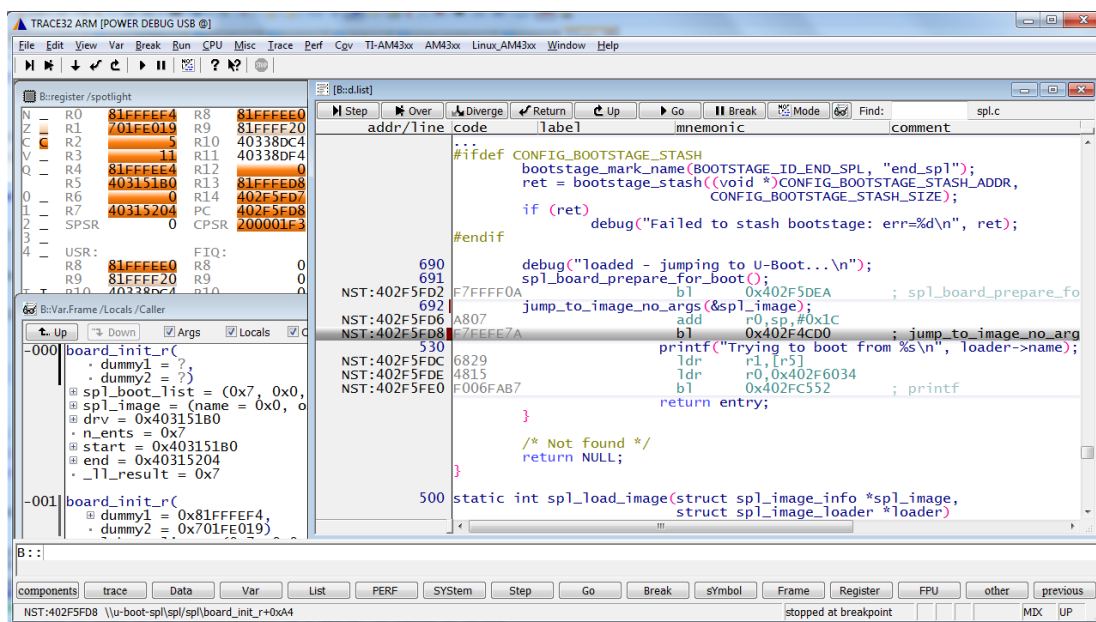
f) (arch/arm/mach-omap2/boot-common.c)

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

The screenshot displays the TRACE32 ARM debugger interface. The main window shows the disassembly of the code starting at address 0x8080000. The register window on the left shows the values of the registers, including R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, and the SPSR. The command window at the bottom shows the command 'NSR:80800000 \\u-boot\\global_image_copy_start' and the status 'stopped at breakpoint'.

Register Window (Left):

Register	Value
R0	40338DC4
R1	701FE019
R2	40337A04
R3	80800000
R4	40325B9C
R5	0
R6	81FFFF00
R7	40325B9C
R8	0
R9	81FFFF20
R10	0
R11	0
R12	81FFFF9C
R13	0
R14	0
SPSR	0

Disassembly Window (Main):

Address	Line	Code	Label	Mnemonic	Comment
NSR:80800000	54	EA00000E	reset	reset	
NSR:80800004	55	E59FF014	ldr	pc, _undefined_instruction	; pc, _undefined_instruction
NSR:80800008	56	E59FF014	ldr	pc, _software_interrupt	; pc, _software_interrupt
NSR:8080000C	57	E59FF014	ldr	pc, _prefetch_abort	; pc, _prefetch_abort
NSR:80800010	58	E59FF014	ldr	pc, _data_abort	; pc, _data_abort
NSR:80800014	59	E59FF014	ldr	pc, _not_used	; pc, _not_used
NSR:80800018	60	E59FF014	ldr	pc, _irq	; pc, _irq
NSR:8080001C	61	E59FF014	ldr	pc, _fiq	; pc, _fiq
NSP:80800020	80800060	..undefin.:dcd	0x80800060	..undefin.:dcd	..undefin.:dcd
NSP:80800024	808000C0	..softwar.:dcd	0x808000C0	..softwar.:dcd	..softwar.:dcd
NSP:80800028	80800120	..prefetc.:dcd	0x80800120	..prefetc.:dcd	..prefetc.:dcd
NSP:8080002C	80800180	..data_ab.:dcd	0x80800180	..data_ab.:dcd	..data_ab.:dcd
NSP:80800030	808001E0	..not_used:dcd	0x808001E0	..not_used:dcd	..not_used:dcd
NSP:80800034	80800240	..irq: dcd	0x80800240	..irq: dcd	..irq: dcd
NSP:80800038	808002A0	..fiq: dcd	0x808002A0	..fiq: dcd	..fiq: dcd
NSP:8080003C	DEADBEEF	..dcd	0xDEADBEEF	..dcd	..dcd
NSP:80800040	0BADCODE	..dcd	0x0BADCODE	..dcd	..dcd
NSR:80800044	E32F0000	nop		nop	
NSR:80800048	E32F0000	nop		nop	
NSR:8080004C	E32F0000	nop		nop	
NSR:80800050	E32F0000	nop		nop	
NSR:80800054	E32F0000	nop		nop	

Command Window (Bottom):

Components: trace Data Var List PERF SYSTEM Step Go Break Symbol Frame Register FPU other previous

NSR:80800000 \\u-boot\\global_image_copy_start

stopped at breakpoint

MDX UP