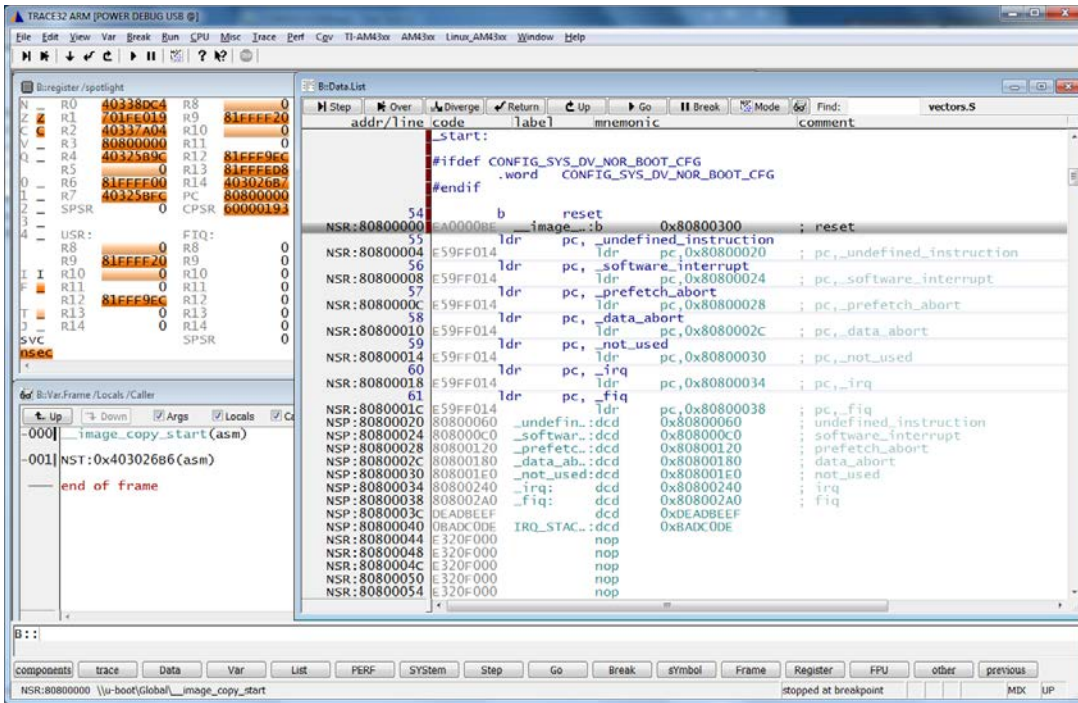
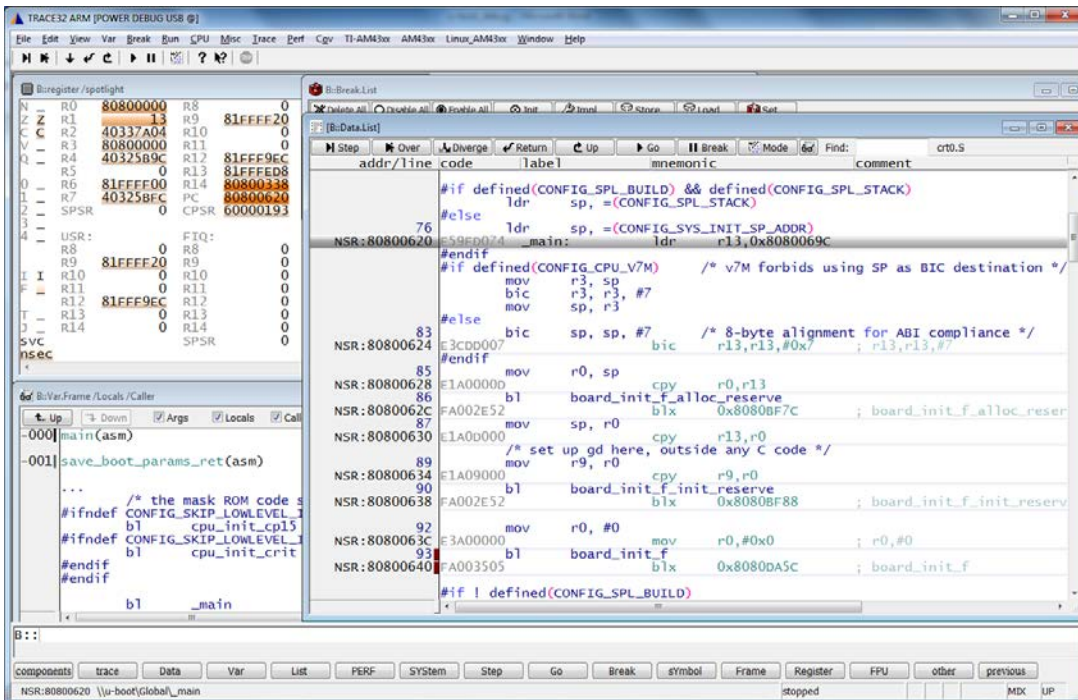


# How to JTAG debug u-boot with relocation

1. At the u-boot entry.



2. At the entry of ENTRY(\_main) in crt0.S for C-runtime startup Code for ARM U-Boot.



```

/*
 * 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. GDs 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.
 */

```

Note that the very first SPL serial output as shown below is from board\_init\_f().

```

U-Boot 2017.01-00319-g42b46bd-dirty (Aug 05 2020 - 12:17:33 -0700)
CPU : AM437X-HS rev 1.2
Model: TI AM437x GP EVM
DRAM: 2 GiB

```

### 3. Set bkpt @relocate\_code:

The screenshot shows the TRACE32 ARM debugger interface. The Breakpoint List window is open, showing a breakpoint set at address 80800664, which is the start of the `relocate_code` function. The Data List window shows the assembly code for `relocate_code`, starting with `ldr r0, [r9, #GD_RELOC_OFF]` and `ldr r0, [r9, #0x40]`. The Register/spotlight window shows the current register values, including R0 at 80800664 and R14 at FFF29668. The Call stack window shows the current frame for `main(asm)`.

### 4. Step into relocate\_code.

The relocation offset is in R4=0x7F729000 in this example

The screenshot shows the TRACE32 ARM debugger interface with the breakpoint hit. The Data List window shows the assembly code for `relocate_code`, starting with `ENTRY(relocate_code)` and `ldr r1, =_image_copy_start`. The Register/spotlight window shows the current register values, including R4 at 7F729000 and R14 at FFF29668. The Call stack window shows the current frame for `relocate_code(asm)`.

5. At the exit of `relocate_code`, this is the last instruction of u-boot running w/o relocation.

The screenshot shows the TRACE32 ARM debugger interface. The register window on the left displays the state of registers R0 through R14, with R7 (PC) at 40325BFC and R10 (808604C0) highlighted. The B:Break.List window shows a list of breakpoints, with NSR:80800A9C (Program SOFT) at address 80800A9C labeled 'relocate\_done' highlighted. The B:Data.List window shows the disassembly of the `relocate_code` function, with the instruction `ldr r1, =_image_copy_start /* r1 <- SRC & _image_copy_start */` at address 81 highlighted. The stack frame window shows the `relocate_done(asm)` function call.

6. One step further, u-boot running after relocation

The screenshot shows the TRACE32 ARM debugger interface after relocation. The register window on the left displays the state of registers R0 through R14, with R7 (PC) at FFF29668 and R10 (808604C0) highlighted. The B:Break.List window shows a list of breakpoints, with NSR:80800A9C (Program SOFT) at address 80800A9C labeled 'relocate\_done' highlighted. The B:Data.List window shows the disassembly of the code after relocation, with the instruction `ldr r14, 0xFFFF29668` at address NSR:FFF29654 highlighted. The stack frame window shows the `relocate_done(asm)` function call.

- Relocate the u-boot symbol using <y.reloc 0x7F729000> for T32 JTAG debugger. The screenshot in #6 is shown below which matches u-boot SRC nicely.

The screenshot displays the TRACE32 ARM debugger interface with the following components:

- Register Window (B:register /spotlight):** Shows register values for R0 through R14 and CPSR. R10 is highlighted with the value 808604C0.
- Break List (B:Break.List):** A table listing break points with columns for address, type, impl, and store/load status.
- Data List (B:Data.List):** Shows assembly code for crt0.S with columns for address/line, code, label, mnemonic, and comment. Line 127 is highlighted.
- Variable Window (B:Var.Frame /Locals /Caller):** Shows local variables like 'here' and 'relocate\_code'.

**Break List Data:**

address	types	impl	store/load
NR:80800000	Program	ONCHIP	NR:0x80800000
NR:80800334	Program	ONCHIP	NR:0x80800334
NR:80800664	Program	ONCHIP	NR:0x80800664
NR:80800A9C	Program	SOFT	NR:0x80800A9C

**Data List Code Snippet:**

```

NSR:FFF2965C E08EE000 add r14,r14,r0
/* if defined(CONFIG_CPU_V7M)
   orr lr, #1 /* As required by Thumb-only
#endif
NSR:FFF29660 E599002C ldr r0, [r9, #GD_RELOCADDR] /* r0 = gd->relocaddr */
NSR:FFF29664 EA0000F8 b relocate_code
here:
/* now relocate vectors
*/
NSR:FFF29668 EB0000F4 here: b 0xFFFF29A40 ; relocate_vectors
/* Set up final (full) environment */
NSR:FFF2966C EBFFFF31 b1 c_runtime_cpu_setup /* we still call old routine here */
EBFFFF31 b1 0xFFFF29338 ; c_runtime_cpu_setup
#endif
#if !defined(CONFIG_SPL_BUILD) || defined(CONFIG_SPL_FRAMEWORK)
# ifdef CONFIG_SPL_BUILD
/* use a DRAM stack for the rest of SPL, if requested */
b1 spl_relocate_stack_gd

```