Memory Related Parameters Passing in TI Keystone II Linux Kernel and U-Boot

Recently, customer submitted questions about how to modify the memory size and location in Linux kernel and U-Boot, based on current understanding, let's discuss about the memory related parameters passing in Linux kernel and U-boot.

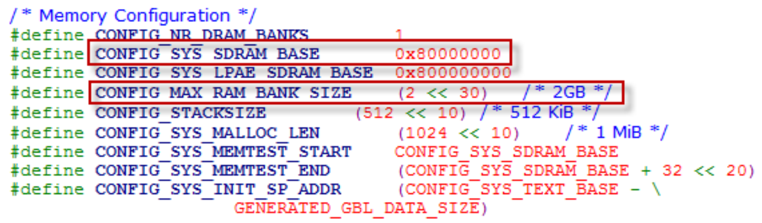
We use the Linux kernel (K2_LINUX_03.08.04_13.09), Boot monitor (K2_BM_13.08) and U-Boot (K2_UBOOT_2013-01_13.09) source code on Arago site as example for the following parts.

1. Memory Info, Parsing and Passing in U-Boot

U-Boot holds the memory info and initializes the memory part in initialization procedures.

1.1 Initial Memory Info

In Keystone II U-Boot, we have the board specified header file in path of ${U_BOOT_KEYSTONE}/include/configs/tci6638_evm.h defined the initial memory physical start address and limit size for system initialization:

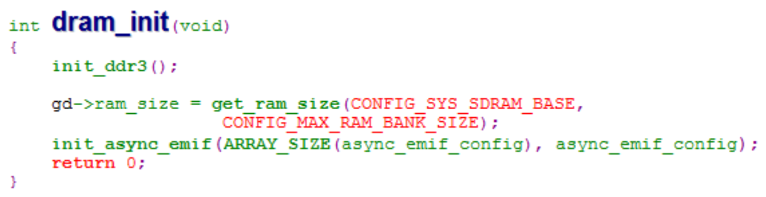


```
/* Memory Configuration */
#define CONFIG_NR_DRAM_BANKS            1
#define CONFIG_SYS_SDRAM_BASE           0x80000000
#define CONFIG_SYS_LPAE_SDRAM_BASE      0x800000000
#define CONFIG_MAX_RAM_BANK_SIZE        (2 << 30)      /* 2GB */
#define CONFIG_STACKSIZE            (512 << 10) /* 512 KiB */
#define CONFIG_SYS_MALLOC_LEN           (1024 << 10)      /* 1 MiB */
#define CONFIG_SYS_MEMTEST_START       CONFIG_SYS_SDRAM_BASE
#define CONFIG_SYS_MEMTEST_END          (CONFIG_SYS_SDRAM_BASE + 32 << 20)
#define CONFIG_SYS_INIT_SP_ADDR         (CONFIG_SYS_TEXT_BASE - \
                    GENERATED_GBL_DATA_SIZE)
```

1.2 Memory Parameters Parsing and Passing Procedures

1.2.1 Memory Parameters Parsing

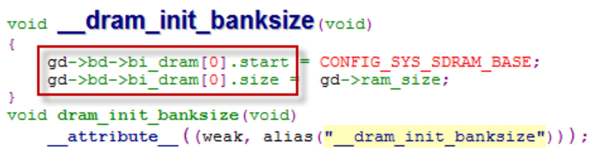
During the U-Boot hardware initialization function "board_init_f" (${U_BOOT_KEYSTONE}/arch/arm/lib/board.c), the sequential initialization loop will call function "dram_init" (${U_BOOT_KEYSTONE}/board/ti/tci6638_evm/board.c) for parsing the memory info.



```
int dram_init(void)
{
    init_ddr3();

    gd->ram_size = get_ram_size(CONFIG_SYS_SDRAM_BASE,
                    CONFIG_MAX_RAM_BANK_SIZE);
    init_async_emif(ARRAY_SIZE(async_emif_config), async_emif_config);
    return 0;
}
```

In this function, after initialize the DDR controller, it'll calculate the actual memory size based on simple memory access tests, then U-Boot get the actual valid memory size installed on hardware.

In later of "board_init_f", the function "dram_init_banksize" (${U_BOOT_KEYSTONE}/arch/arm/lib/board.c) parse and get the actual physical memory start address and valid installed memory size for later usage:



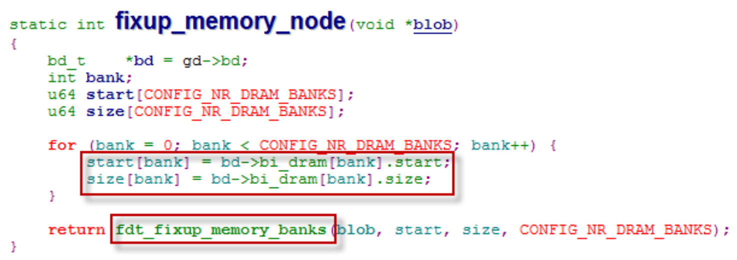
```
void __dram_init_banksize(void)
{
    gd->bd->bi_dram[0].start = CONFIG_SYS_SDRAM_BASE;
    gd->bd->bi_dram[0].size =  gd->ram_size;
}
void dram_init_banksize(void)
    __attribute__((weak, alias("__dram_init_banksize")));
```

The updated memory info is saved in variables "gd->bd->bi_dram[0].start" & "gd->bd->bi_dram[0].size".

1.2.2 Memory Parameters Update and Passing

After the U-Boot hardware initialization, when receive the "bootm" command from console, U-Boot will call the function "do_bootm_linux" (${U_BOOT_KEYSTONE}/arch/arm/lib/bootm.c), this function will call nested function "do_bootm_linux" -> "boot_prep_linux" -> "create_fdt" to update the compiled DTB file with actual or U-Boot used parameters the pass the updated DTB file to Linux kernel.

The function "fixup_memory_node" (${U_BOOT_KEYSTONE}/arch/arm/lib/bootm.c) update the DTB with the memory info:

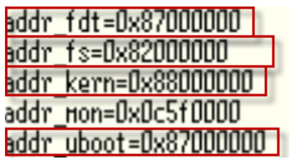


```
static int fixup_memory_node(void *blob)
{
    bd_t    *bd = gd->bd;
    int bank;
    u64 start[CONFIG_NR_DRAM_BANKS];
    u64 size[CONFIG_NR_DRAM_BANKS];

    for (bank = 0; bank < CONFIG_NR_DRAM_BANKS; bank++) {
        start[bank] = bd->bi_dram[bank].start;
        size[bank] = bd->bi_dram[bank].size;
    }

    return fdt_fixup_memory_banks(blob, start, size, CONFIG_NR_DRAM_BANKS);
}
```

It'll use the saved memory info variables to update the memory related items in DTB file

1.3 Other Parameters based on the Initial Memory Info

In U-Boot command environmental variables, some parameters based on the start address in memory info, these parameters are:



```
addr_fdt=0x87000000
addr_fs=0x82000000
addr_kern=0x88000000
addr_mon=0x0c5f0000
addr_uboot=0x87000000
```

When you want to modify the memory info especially for the memory start address, please also make sure to update these variables in U-Boot via command such as:

```
setenv addr_kern '0xC0000000'
```

Then save the modification back to NAND flash by using:

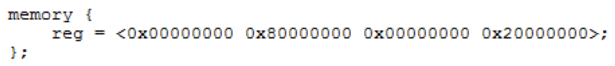
```
saveenv
```

2. Memory Info, Parsing and Using in Linux Kernel

2.1 Initial Memory Info

Due to Keystone II Linux kernel fully support Flat Device Tree architecture, the main hardware related info are in DTS file, which the compiled binary DTB file will be passed to Linux kernel through U-Boot.

By checking the DTS file, we can easily find the initial memory info such as follow:



```
memory {
      reg = <0x00000000 0x80000000 0x00000000 0x20000000>;
};
```

This section defined the start address of DDR is started from 0x8000_0000 and length is 0x2000_0000 (512MByte)

2.2 Memory Parameters Parsing and Usage Procedures

In Linux kernel initialization part, the function “setup_arch” (${LINUX_KEYSTONE}/arch/arm/kernel/setup.c) mainly setup architecture related functional call for hardware specified parts.

It will call function “setup_machine_fdt” (${LINUX_KEYSTONE}/arch/arm/kernel/devtree.c) for the DTB file parsing procedures.

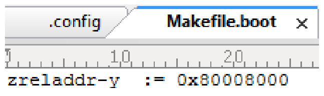
The function “early_init_dt_scan_memory” (${LINUX_KEYSTONE}/drivers/of/fdt.c) called by “setup_machine_fdt” will get the specific memory info and save in relative variables:

```
int __init early_init_dt_scan_memory(unsigned long node, const char *uname,
                                     int depth, void *data)
{
    char *type = of_get_flat_dt_prop(node, "device_type", NULL);
    __be32 *reg, *endp;
    unsigned long l;

    /* We are scanning "memory" nodes only */
    if (type == NULL) {
        /*
         * The longtrail doesn't have a device_type on the
         * /memory node, so look for the node called /memory@0.
         */
        if (depth != 1 || strcmp(uname, "memory@0") != 0)
            return 0;
    } else if (strcmp(type, "memory") != 0)
        return 0;

    reg = of_get_flat_dt_prop(node, "linux,usable-memory", &l);
    if (reg == NULL)
        reg = of_get_flat_dt_prop(node, "reg", &l);
    if (reg == NULL)
        return 0;

    endp = reg + (l / sizeof(__be32));

    pr_debug("memory scan node %s, reg size %ld, data: %x %x %x %x,\n",
        uname, l, reg[0], reg[1], reg[2], reg[3]);

    while ((endp - reg) >= (dt_root_addr_cells + dt_root_size_cells)) {
        u64 base, size;

        base = dt_mem_next_cell(dt_root_addr_cells, &reg);
        size = dt_mem_next_cell(dt_root_size_cells, &reg);

        if (size == 0)
            continue;
        pr_debug(" - %llx ,  %llx\n", (unsigned long long)base,
            (unsigned long long)size);

        early_init_dt_add_memory_arch(base, size);
    }

    return 0;
} ? end early_init_dt_scan_memory ?
```

2.3 Other Parameters based on the Initial Memory Info

Most variables in Linux kernel which uses the initialization memory info will base the initial value which parsed from the DTB file. But some variables use the fixed base address which need extra modification when changing the initial memory base address.

The makefile saved relocation address based on the physical memory start address plus offsets, if you modify the memory physical start address, please make sure also change the relocation address



```
zreladdr-y   := 0x80008000
```

3. Memory Info Modification Notes

User sometimes needs to modify the memory info to meet their memory allocation requirements or hardware board schematics.

To modify the memory info in system, only modify the DTS will not be enough, during the previous introduction, we know that U-Boot will update the DTS file to make the hardware information accurate and valid, **please aware that the DTB which U-Boot passes to Linux kernel may not the same with the original DTB file.** Extra modification also needs to be applied in U-Boot memory info parts.

The parameters based on initial memory info which mentioned in previous sections will also need to be modified to your desired values