```
1.  int image_setup_linux(bootm_headers_t *images)
2.  {
3.          ulong of_size = images->ft_len;
4.          char **of_flat_tree = &images->ft_addr;
5.          ulong *initrd_start = &images->initrd_start;
6.          ulong *initrd_end = &images->initrd_end;
7.          struct lmb *lmb = &images->lmb;
8.          ulong rd_len;
9.          int ret;
10.
11.         if (IMAGE_ENABLE_OF_LIBFDT)
12.                 boot_fdt_add_mem_rsv_regions(lmb, *of_flat_tree);
                ①
13.
14.         if (IMAGE_BOOT_GET_CMDLINE) {
15.                 ret = boot_get_cmdline(lmb, &images->cmdline_start,
            ②
16.                                        &images->cmdline_end);
17.                 if (ret) {
18.                         puts("ERROR with allocation of cmdline\n");
19.                         return ret;
20.                 }
21.         }
22.         if (IMAGE_ENABLE_RAMDISK_HIGH) {
23.                 rd_len = images->rd_end - images->rd_start;
                            ③
24.                 ret = boot_ramdisk_high(lmb, images->rd_start, rd_len,
25.                                         initrd_start, initrd_end);
26.                 if (ret)
27.                         return ret;
28.         }
29.
30.         if (IMAGE_ENABLE_OF_LIBFDT) {
31.                 ret = boot_relocate_fdt(lmb, of_flat_tree, &of_size);
        ④
32.                 if (ret)
33.                         return ret;
34.         }
35.
36.         if (IMAGE_ENABLE_OF_LIBFDT && of_size) {
37.                 ret = image_setup_libfdt(images, *of_flat_tree, of_size, lmb);
            ⑤
38.                 if (ret)
39.                         return ret;
40.         }
41.
42.         return 0;
43. }
```

①

从dtb中获得memreserve的信息。

```
/**
 * boot_fdt_add_mem_rsv_regions - Mark the memreserve sections as unusable
 * @lmb: pointer to lmb handle, will be used for memory mgmt
 * @fdt_blob: pointer to fdt blob base address
 *
 * Adds the memreserve regions in the dtb to the lmb block.  Adding the
 * memreserve regions prevents u-boot from using them to store the initrd
 * or the fdt blob.
 */
void boot_fdt_add_mem_rsv_regions(struct lmb *lmb, void *fdt_blob)
{
        uint64_t addr, size;
        int i, total;

        if (fdt_check_header(fdt_blob) != 0)
                return;

        total = fdt_num_mem_rsv(fdt_blob);
        for (i = 0; i < total; i++) {
                if (fdt_get_mem_rsv(fdt_blob, i, &addr, &size) != 0)
                        continue;
                printf("   reserving fdt memory region: addr=%llx size=%llx\n",
                       (unsigned long long)addr, (unsigned long long)size);
                lmb_reserve(lmb, addr, size);
        }
}
```

根据ePAPR Version 1.1 - 08 April 2011

Memory reservations define an entry for the device tree blob's memory reservation table.

They have the form:

e.g., /memreserve/ <address> <length>;

Where <address> and <length> are 64-bit C-style integers.

比如：

linux/arch/arm/boot/dts/ecx-2000.dts

```
1.  /memreserve/ 0x00000000 0x0001000;
```

axm5516-amarillo.dts

```
1.  /memreserve/ 0x00000000 0x00400000;
```

boot_fdt_add_mem_rsv_regions()获得dtb中/memreserve/ section中的(address length) pair,

然后记录在lmb (Logic Memory Block)的reserved region。在后面的step⑤会用到。

②

in G2 LSP, IMAGE_BOOT_GET_CMDLINE is false.

```
#ifdef CONFIG_SYS_BOOT_GET_CMDLINE
/**
 * boot_get_cmdline - allocate and initialize kernel cmdline
 * @lmb: pointer to lmb handle, will be used for memory mgmt
 * @cmd_start: pointer to a ulong variable, will hold cmdline start
 * @cmd_end: pointer to a ulong variable, will hold cmdline end
 *
 * boot_get_cmdline() allocates space for kernel command line below
 * BOOTMAPSZ + getenv_bootm_low() address. If "bootargs" U-boot environemnt
 * variable is present its contents is copied to allocated kernel
 * command line.
 *
 * returns:
 *      0 - success
 *     -1 - failure
 */
int boot_get_cmdline(struct lmb *lmb, ulong *cmd_start, ulong *cmd_end)
{
        char *cmdline;
        char *s;

        cmdline = (char *)(ulong)lmb_alloc_base(lmb, CONFIG_SYS_BARGSIZE, 0xf,
                                getenv_bootm_mapsize() + getenv_bootm_low());

        if (cmdline == NULL)
                return -1;

        if ((s = getenv("bootargs")) == NULL)
                s = "";

        strcpy(cmdline, s);

        *cmd_start = (ulong) & cmdline[0];
        *cmd_end = *cmd_start + strlen(cmdline);

        debug("## cmdline at 0x%08lx ... 0x%08lx\n", *cmd_start, *cmd_end);

        return 0;
}
#endif /* CONFIG_SYS_BOOT_GET_CMDLINE */
```

boot_get_cmdline()从"bootargs" environment variable中获取kernel的boot parameter,然后记录在bootm_headers_t structure中(　　cmdline_start,cmdline_end fields)，但好像没用到。

③

由于G2 LSP没有用到initramfs,这里不会调用boot_ramdisk_high()。

boot_ramdisk_high()把initramfs move到内存高端，可能是为了给kernel尽量留出连续的memory(可能这也是u-boot本身

被relocate到物理内存顶端的缘故)

④

boot_relocate_fdt()意图同step ③一样，为了给kernel留出连续的低端内存，想把dtb move到内存高端。是否move，要看是否

定义了"fdt_high" environment variable。G2 LSP中没有定义，所以dtb并没有被move，还是在0xf00000。

```
1.  /**
2.   * boot_relocate_fdt - relocate flat device tree
3.   * @lmb: pointer to lmb handle, will be used for memory mgmt
4.   * @of_flat_tree: pointer to a char* variable, will hold fdt start address
5.   * @of_size: pointer to a ulong variable, will hold fdt length
6.   *
7.   * boot_relocate_fdt() allocates a region of memory within the bootmap and
8.   * relocates the of_flat_tree into that region, even if the fdt is already in
9.   * the bootmap.  It also expands the size of the fdt by CONFIG_SYS_FDT_PAD
10.  * bytes.
11.  *
12.  * of_flat_tree and of_size are set to final (after relocation) values
13.  *
14.  * returns:
15.  *      0 - success
16.  *      1 - failure
17.  */
18. int boot_relocate_fdt(struct lmb *lmb, char **of_flat_tree, ulong *of_size)
19. {
20.         void    *fdt_blob = *of_flat_tree;
21.         void    *of_start = NULL;
22.         char    *fdt_high;
23.         ulong   of_len = 0;
24.         int     err;
25.         int     disable_relocation = 0;
26.
27.         /* nothing to do */
28.         if (*of_size == 0)
29.                 return 0;
30.
31.         if (fdt_check_header(fdt_blob) != 0) {
32.                 fdt_error("image is not a fdt");
33.                 goto error;
34.         }
35.
36.         /* position on a 4K boundary before the alloc_current */
37.         /* Pad the FDT by a specified amount */
38.         of_len = *of_size + CONFIG_SYS_FDT_PAD;
39.
40.         /* If fdt_high is set use it to select the relocation address */
41.         fdt_high = getenv("fdt_high");

                        (A)
42.         if (fdt_high) {
43.                 void *desired_addr = (void *)simple_strtoul(fdt_high, NULL, 16);
44.
45.                 if (((ulong) desired_addr) == ~0UL) {

        (B)
46.                         /* All ones means use fdt in place */
47.                         of_start = fdt_blob;
48.                         lmb_reserve(lmb, (ulong)of_start, of_len);
```

```
(C)
                        disable_relocation = 1;

                                                (D)
                } else if (desired_addr) {
                        of_start =
                            (void *)(ulong) lmb_alloc_base(lmb, of_len, 0x1000,
                                                           (ulong)desired_addr);
                        if (of_start == NULL) {
                                puts("Failed using fdt_high value for Device Tree
");
                                goto error;
                        }
                } else {
                        of_start =
                            (void *)(ulong) lmb_alloc(lmb, of_len, 0x1000);
                }
        } else {
                of_start =
                    (void *)(ulong) lmb_alloc_base(lmb, of_len, 0x1000,
                                                   getenv_bootm_mapsize()
                                                   + getenv_bootm_low());
        }

        if (of_start == NULL) {
                puts("device tree - allocation error\n");
                goto error;
        }

        if (disable_relocation) {

                                                (E)
                /*
                 * We assume there is space after the existing fdt to use
                 * for padding
                 */
                fdt_set_totalsize(of_start, of_len);
                printf("   Using Device Tree in place at %p, end %p\n",
                       of_start, of_start + of_len - 1);
        } else {
                debug("## device tree at %p ... %p (len=%ld [0x%lX])\n",
                      fdt_blob, fdt_blob + *of_size - 1, of_len, of_len);

                printf("   Loading Device Tree to %p, end %p ... ",
                       of_start, of_start + of_len - 1);

                err = fdt_open_into(fdt_blob, of_start, of_len);
                if (err != 0) {
                        fdt_error("fdt move failed");
                        goto error;
                }
                puts("OK\n");
        }

```

```
97.             *of_flat_tree = of_start;
98.             *of_size = of_len;
99.
100.            set_working_fdt_addr(*of_flat_tree);
101.            return 0;
102.
103.    error:
104.            return 1;
105.    }
```

(A)

in board/pegmatite/setup.c

```
1.    setenv("fdt_high", "ffffffff");
```

getenv("fdt_high") return non-NULL.

(B)

~0 == ffffffff

(C)

把dtb所占space记录在logical block memory的reserved region。

(D)

disable_relocation = 1,置位。

(E)

下面来自G2 LSP u-boot的log

```
1.      Using Device Tree in place at 00f00000, end 00f140c6
```

⑤

image_setup_libfdt() function,u-boot修改dtb。

```
1.  int image_setup_libfdt(bootm_headers_t *images, void *blob,
2.                         int of_size, struct lmb *lmb)
3.  {
4.          ulong *initrd_start = &images->initrd_start;
5.          ulong *initrd_end = &images->initrd_end;
6.          int ret;
7.
8.          if (fdt_chosen(blob, 1) < 0) {
                                                              (A)
9.                  puts("ERROR: /chosen node create failed");
10.                 puts(" - must RESET the board to recover.\n");
11.                 return -1;
12.         }
13.         arch_fixup_memory_node(blob);
                                                          (B)
14.         if (IMAAGE_OF_BOARD_SETUP)
15.                 ft_board_setup(blob, gd->bd);
                                                              (C)
16.         fdt_fixup_ethernet(blob);


17.
18.         /* Delete the old LMB reservation */
19.         lmb_free(lmb, (phys_addr_t)(u32)(uintptr_t)blob,
20.                  (phys_size_t)fdt_totalsize(blob));
21.
22.         ret = fdt_resize(blob);
23.         if (ret < 0)
24.                 return ret;
25.         of_size = ret;
26.
27.         if (*initrd_start && *initrd_end) {
28.                 of_size += FDT_RAMDISK_OVERHEAD;
29.                 fdt_set_totalsize(blob, of_size);
30.         }
31.         /* Create a new LMB reservation */
32.         lmb_reserve(lmb, (ulong)blob, of_size);
33.
34.         fdt_initrd(blob, *initrd_start, *initrd_end, 1);
                       (D)
35.         if (!ft_verify_fdt(blob))
36.                 return -1;
37.
38.         return 0;
39. }
```

(A)

u-boot修改原来dtb中的bootargs就在该function中实现。

```
int fdt_chosen(void *fdt, int force)
{
        int   nodeoffset;
        int   err;
        char  *str;             /* used to set string properties */
        const char *path;

        err = fdt_check_header(fdt);
        if (err < 0) {
                printf("fdt_chosen: %s\n", fdt_strerror(err));
                return err;
        }

        /*
         * Find the "chosen" node.
         */
        nodeoffset = fdt_path_offset (fdt, "/chosen");

        /*
         * If there is no "chosen" node in the blob, create it.
         */
        if (nodeoffset < 0) {
                /*
                 * Create a new node "/chosen" (offset 0 is root level)
                 */
                nodeoffset = fdt_add_subnode(fdt, 0, "chosen");
                if (nodeoffset < 0) {
                        printf("WARNING: could not create /chosen %s.\n",
                                fdt_strerror(nodeoffset));
                        return nodeoffset;
                }
        }

        /*
         * Create /chosen properites that don't exist in the fdt.
         * If the property exists, update it only if the "force" parameter
         * is true.
         */
        str = getenv("bootargs");
if (str != NULL) {
                path = fdt_getprop(fdt, nodeoffset, "bootargs", NULL);
if ((path == NULL) || force) {
                        err = fdt_setprop(fdt, nodeoffset,
"bootargs", str, strlen(str)+1);
if (err < 0)
                                printf("WARNING: could not set bootargs %s.\n",
                                        fdt_strerror(err));
}
}

#ifdef CONFIG_OF_STDOUT_VIA_ALIAS
        path = fdt_getprop(fdt, nodeoffset, "linux,stdout-path", NULL);
        if ((path == NULL) || force)
```

```
                err = fdt_fixup_stdout(fdt, nodeoffset);
#endif

#ifdef OF_STDOUT_PATH
        path = fdt_getprop(fdt, nodeoffset, "linux,stdout-path", NULL);
        if ((path == NULL) || force) {
                err = fdt_setprop(fdt, nodeoffset,
                        "linux,stdout-path", OF_STDOUT_PATH, strlen(OF_STDOUT_PAT
H)+1);
                if (err < 0)
                        printf("WARNING: could not set linux,stdout-path %s.\n",
                                fdt_strerror(err));
        }
#endif

        return err;
}
```

(B)

in arch/arm/lib/bootm-fdt.c

```
int arch_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);
}
```

in common/fdt_support.c

```c
#ifdef CONFIG_NR_DRAM_BANKS
#define MEMORY_BANKS_MAX CONFIG_NR_DRAM_BANKS
#else
#define MEMORY_BANKS_MAX 4
#endif
int fdt_fixup_memory_banks(void *blob, u64 start[], u64 size[], int banks)
{
        int err, nodeoffset;
        int addr_cell_len, size_cell_len, len;
        u8 tmp[MEMORY_BANKS_MAX * 16]; /* Up to 64-bit address + 64-bit size */
        int bank;

        if (banks > MEMORY_BANKS_MAX) {
                printf("%s: num banks %d exceeds hardcoded limit %d."
                       " Recompile with higher MEMORY_BANKS_MAX?\n",
                       __FUNCTION__, banks, MEMORY_BANKS_MAX);
                return -1;
        }

        err = fdt_check_header(blob);
        if (err < 0) {
                printf("%s: %s\n", __FUNCTION__, fdt_strerror(err));
                return err;
        }

        /* update, or add and update /memory node */
        nodeoffset = fdt_path_offset(blob, "/memory");
        if (nodeoffset < 0) {
                nodeoffset = fdt_add_subnode(blob, 0, "memory");
                if (nodeoffset < 0) {
                        printf("WARNING: could not create /memory: %s.\n",
                                        fdt_strerror(nodeoffset));
                        return nodeoffset;
                }
        }
        err = fdt_setprop(blob, nodeoffset, "device_type", "memory",
                        sizeof("memory"));
        if (err < 0) {
                printf("WARNING: could not set %s %s.\n", "device_type",
                                fdt_strerror(err));
                return err;
        }

        addr_cell_len = get_cells_len(blob, "#address-cells");
        size_cell_len = get_cells_len(blob, "#size-cells");

        for (bank = 0, len = 0; bank < banks; bank++) {
                write_cell(tmp + len, start[bank], addr_cell_len);
                len += addr_cell_len;

                write_cell(tmp + len, size[bank], size_cell_len);
                len += size_cell_len;
        }
```

```
54.
55.             err = fdt_setprop(blob, nodeoffset, "reg", tmp, len);
56.             if (err < 0) {
57.                     printf("WARNING: could not set %s %s.\n",
58.                                     "reg", fdt_strerror(err));
59.                     return err;
60.             }
61.             return 0;
62.     }
```

in include/configs/pegmatite.h

```
1.  /*-----------------------------------------------------------------------
2.   * Physical Memory Map
3.   */
4.  #define CONFIG_MC_BASE          0xd0000000
5.  #define CONFIG_MC_BANKS         3
6.  #define CONFIG_NR_DRAM_BANKS    (CONFIG_MC_BANKS + 1)
7.  #define CONFIG_SYS_SDRAM_BASE   0
```

fdt_fixup_memory_banks()就是把bd->bi_dram[]中的memory info写到dtb的/memory/ node中。

一般/memory/ node在dts中由如下format:

memory {

device_type = "memory";

reg = <0x00000000 0x20000000>;

};

fdt_fixup_memory_banks() function parse /memory/ node format。

(C)

in board/pegmatite/setup.c

```
#ifdef CONFIG_OF_BOARD_SETUP
void ft_board_setup(void *blob, bd_t *bd)
{
    uint64_t dram_start[CONFIG_NR_DRAM_BANKS+1], dram_size[CONFIG_NR_DRAM_BANKS+1
];
    int i, j = 0;

    for (i = 0; i < CONFIG_NR_DRAM_BANKS; ++i) {
        dram_start[j] = usable_banks[i].start;

        if ((usable_banks[i].start < 0x80000000)
            && ((usable_banks[i].start + usable_banks[i].size) > 0x80000000)) {
            dram_size[j] = 0x80000000 - usable_banks[i].start;
            ++j;

            dram_start[j] = 0x80000000ULL;
            dram_size[j] = usable_banks[i].size - 0x80000000;
        }
        else {
            dram_size[j] = usable_banks[i].size;
        }

        if (dram_start[j] >= 0x80000000)
            dram_start[j] |= ((uint64_t)1 << 35);

        ++j;
    }

    fdt_fixup_memory_banks(blob, dram_start, dram_size, j);
    ft_lcd_setup(blob, bd);
}
#endif
```

把pegmatite board上的DDR RAM状况反映到dtb中去。

(D)

G2 LSP没有用到initrd (initramfs)。