uboot 显示 logo 之 AM335X

一、uboot 版本：uboot-2013.10

二、需要有 LCD 的驱动支持，TI 已经提供：drivers/video/da8xx-fb.c

三、在 config 头文件（include/configs/am335x_evm.h）中定义相关的宏：

```
/* djf 20151125 add for uboot logo */

#define CONFIG_VIDEO
#define CONFIG_VIDEO_DA8XX // am335x 的 LCD 驱动
#define CONFIG_CFB_CONSOLE
#define CONFIG_CMD_BMP
#define DA8XX_LCD_CNTL_BASE        0x4830E000 // am335x 的 LCD 寄存器基地址
#define CONFIG_SYS_CONSOLE_IS_IN_ENV
#define VIDEO_TSTC_FCT              serial_tstc
#define VIDEO_GETC_FCT              serial_getc
#define VIDEO_KBD_INIT_FCT          0
#define CONFIG_VIDEO_LOGO

#define CONFIG_SPLASH_SCREEN //定义后，才会调用函数显示 nand 或 sd 卡中的图片
/*
#define CONFIG_SPLASH_SCREEN_ALIGN
#define VIDEO_FB_LITTLE_ENDIAN
*/
/* djf 20151125 add for uboot logo */
```

四、代码调用流程：

1.LCD 的参数设置：

<1> arch/arm/lib/board.c->board_init_r()函数调用 stdio_init()函数（/common/stdio.c）：

```
{
    ……
    board_init();   /* Setup chipselects */ //djf 20151126 mark for uboot logo init LCD
    ……
}
```

<2>函数会调用具体板级文件，am335x 为/board/ti/am335x/board. c:

```
{
    gd->bd->bi_boot_params = CONFIG_SYS_SDRAM_BASE + 0x100;
#if defined(CONFIG_NOR) || defined(CONFIG_NAND)
    gpmc_init();
#endif

    /* djf 20151125 add for uboot logo */
    struct cm_perpll *const cmper = (struct cm_perpll *)CM_PER;
    struct cm_dpll *cmdpll = (struct cm_dpll *)CM_DPLL;
    struct dpll_params dpll_lcd = {24, 1, -1, -1, -1, -1, -1};
```

```
        uint32_t *const clk_domains[] = {
            &cmper->lcdclkctrl,
            0,
        };
        uint32_t *const clk_modules_explicit_en[] = {
            &cmper->lcdclkctrl,
            &cmper->lcdcclkstctrl,
            0,
        };

        do_enable_clocks(clk_domains, clk_modules_explicit_en, 1);
        writel(0x0, &cmdpll->clklcdcpixelclk);
        do_setup_dpll(&dpll_lcd_regs, &dpll_lcd);

         da8xx_video_init(&da8xx_panel_at043, &lcd_cfg, lcd_cfg.bpp);

    /* djf 20151125 add for uboot logo */
    }
```

此函数中需添加参数设置初始化函数（da8xx_video_init(&da8xx_panel_at043, &lcd_cfg, lcd_cfg.bpp);），会调用具体驱动中的函数，此处为/drivers/video/da8xx-fb.c，函数原型如下：

```
    void da8xx_video_init(const struct da8xx_panel *panel,
                          const struct lcd_ctrl_config *lcd_cfg, int bits_pixel)
    {
        lcd_panel = panel;
        da8xx_lcd_cfg = lcd_cfg;
        bits_x_pixel = bits_pixel;
    }
```

由此函数看出需要在板级文件（/board/ti/am335x/board. c）定义好所需的参数结构体，与 linux 中的结构体基本相同，需根据具体 LCD 的参数设置，如下：

```
    /* djf 20151125 add for uboot logo */
    struct dpll_regs dpll_lcd_regs = {
            .cm_clkmode_dpll = CM_WKUP + 0x98,
            .cm_idlest_dpll = CM_WKUP + 0x48,
            .cm_clksel_dpll = CM_WKUP + 0x54,
    };

    #define PLL_GET_M(v)        ((v >> 8) & 0x7FF)
    #define PLL_GET_N(v)        (v & 0X7F)

    int clk_get(int clk)
    {
            uint32_t val;
            uint32_t m, n;
```

```
        val = readl(dpll_lcd_regs.cm_clksel_dpll);
        m = PLL_GET_M(val);
        n = PLL_GET_N(val);
        printf("clk_get val = 0x%x, m = 0x%x, n = 0x%x\n", val, m, n);

        val = m * V_OSCK / n;
        printf("clk = 0x%x V_OSCK = %d\n", val, V_OSCK);
        return (m * V_OSCK) / n;
}

static const struct display_panel disp_panel = {
    QVGA,
    32,
    32,
    COLOR_ACTIVE,
};

static struct da8xx_panel da8xx_panel_at043= {
        .name = "AT070TN94V_1",
    .width = 800,
    .height = 480,
    .hfp = 40,
    .hbp = 40,
    .hsw = 48,
    .vfp = 13,
    .vbp = 29,
    .vsw = 3,
    .pxl_clk = 30000000,
    .invert_pxl_clk = 0,
};

static struct lcd_ctrl_config lcd_cfg = {
    &disp_panel,
    .ac_bias        = 255,
    .ac_bias_intrpt         = 0,
    .dma_burst_sz       = 16,
    .bpp            = 32,
    .fdd            = 255,
    .tft_alt_mode       = 0,
    .stn_565_mode           = 0,
    .mono_8bit_mode     = 0,
    .invert_line_clock  = 1,
    .invert_frm_clock   = 1,
```

```
        .sync_edge          = 0,
        .sync_ctrl      = 1,
        .raster_order       = 0,
    };
    /* djf 20151125 add for uboot logo */
```

其中 clk_get()函数会在/drivers/video/da8xx-fb.c 中 lcd_calc_clk_divider()用到，所以也必须给出。

至此，宏定义及参数设置就算完成了。接下来就是具体驱动的调用及 logo 的显示了。

2.LCD 驱动初始化及 logo 显示

<1>arch/arm/lib/board.c->board_init_r()函数调用 stdio_init()函数（/common/stdio.c）：

```
    {
        ……
        #if defined(CONFIG_VIDEO) || defined(CONFIG_CFB_CONSOLE)
            drv_video_init (); //djf 20151126 mark for uboot logo
        #endif
        ……
    }
```

<2> drv_video_init ()函数在/drivers/video/cfb_console.c 中：

```
    {
        ……
        /* Init video chip - returns with framebuffer cleared */
        skip_dev_init = (video_init() == -1); // djf 20151126 add for uboot logo
        ……
    }
```

<3> 调用同文件下 video_init()函数：

```
    {
        pGD = video_hw_init(); //djf 20151126 mark for uboot logo
        if (pGD == NULL)
            return -1;

        video_fb_address = (void *) VIDEO_FB_ADRS; //djf 20151126 mark for uboot logo :
        framebuffer address(#define VIDEO_FB_ADRS          (pGD->frameAdrs))
    }
```

<4> video_hw_init()函数会调用/drivers/video/下具体驱动内的函数，am335x 的为 da8xx-fb.c。此函数会根据第 1 步在板级文件中设置好的参数及 clk_get()函数对 LCD 进行初始化。

<5>再看<3>中 video_init()函数下面的部分：

```
    {
        #ifdef CONFIG_VIDEO_LOGO
        /* Plot the logo and get start point of console */
        debug("Video: Drawing the logo ...\n");
        video_console_address = video_logo();//djf 20151126 mark for uboot logo
```

```
        #else
            video_console_address = video_fb_address;
        #endif
    }
```

<6>会调用到同文件下的 video_logo()函数：

```
    {
    #ifdef CONFIG_SPLASH_SCREEN
        s = getenv("splashimage"); //djf 20151126 mark for uboot logo
        if (s != NULL) {
            splash_screen_prepare();//什么都没做
            addr = simple_strtoul(s, NULL, 16);

            if (video_display_bitmap(addr,
                        video_logo_xpos,
                        video_logo_ypos) == 0) {
                video_logo_height = 0;
                return ((void *) (video_fb_address));
            } //djf 20151126 mark for uboot logo
        }
    #endif /* CONFIG_SPLASH_SCREEN */
    }
```

此函数中会获取 uboot 环境变量 splashimage 的值（需要在板级 config 文件 /include/configs/am335x_evm.h 中设置，见下图：），

```
#ifndef CONFIG_SPL_BUILD
#define CONFIG_EXTRA_ENV_SETTINGS \
    "loadaddr=0x80200000\0" \
    "kloadaddr=0x80007fc0\0" \
    "fdtaddr=0x80F80000\0" \
    "fdt_high=0xa0000000\0" \
    "boot_fdt=try\0" \
    "rdaddr=0x81000000\0" \
    "bootpart=0:2\0" \
    "bootdir=/ boot\0" \
    "bootfile=ulmage\0" \
    "fdtfile=undefined\0" \
    "console=ttyO0 115200n8\0" \
    "splashimage=0x600000\0" \
    "partitions=" \
        "uuid_disk=${uuid_gpt_disk};" \
        "name=rootfs,start=2MiB,size=- ,uuid=${uuid_gpt_rootfs}\0" \
    "optargs=\0" \
    "dfu_alt_info_mmc=" DFU_ALT_INFO_MMC "\0" \
```

获取后进行转换，这个值就是 bmp 图片的存储地址，但是原生的 uboot 是只支持 Nor Flash 和内核的读取的，如果图片放在 nand flash、SD 卡或者 SPI、I2C 接口 flash 中，需要自己实现将图片文件读取到内存中，然后再显示。调用同文件下 video_display_bitmap()函数实现具体的 logo 显示，函数原型如下：(由于想要显示 nand 或者 sd 卡中的 bmp 图片做为 logo，所以添加了图片地址判断及图片预读取到内存 0x82000000 相关代码，其中涉及到的地址与 am335x 地址分配及板级文件对 nand 分区等有关)

```
    /*
     * Display the BMP file located at address bmp_image.
     */
```

```
int video_display_bitmap(ulong bmp_image, int x, int y)
{
//    printf( "video_display_bitmap() is executing \n" ); //djf 20151126 add for test

      ushort xcount, ycount;
      uchar *fb;
//    bmp_image_t *bmp = (bmp_image_t *) bmp_image; //djf 20151127 del
      bmp_image_t *bmp; //djf 20151127 add
      uchar *bmap;
      ushort padded_line;
      unsigned long width, height, bpp;
      unsigned colors;
      unsigned long compression;
      bmp_color_table_entry_t cte;

#ifdef CONFIG_VIDEO_BMP_GZIP
      unsigned char *dst = NULL;
      ulong len;
#endif

      WATCHDOG_RESET();

      /* djf 20151127 add for uboot logo:copy data to ram from nand */
      char *argv[3];
      char buf[100];


      if(bmp_image < 0x40000000)
      {
            sprintf(buf, "mw.b 0x82000000 0xff 0x200000"); // 清除 2M 内存
            setenv("tempenv", buf);
            argv[0] = "run";
            argv[1] = "tempenv";
            argv[2] = NULL;

            if (do_run(NULL, 0, 2, argv))
            {
                  printf("Error: mw.b err \n");
                  return 1;
            }
            setenv("tempenv", NULL);

            // 读取图片信息到内存
            sprintf(buf, "nand read 0x82000000 0x600000 0x200000");
```

```
            setenv("tempenv", buf);
            argv[0] = "run";
            argv[1] = "tempenv";
            argv[2] = NULL;

            if(do_run(NULL, 0, 2, argv))
            {
                printf("Error: nand read err \n");
                return 1;
            }
            setenv("tempenv", NULL);
            bmp =    (bmp_image_t *)0x82000000;
        }
//      printf("copy logo data from nand done\n"); //djf 20151127 add for test
        /* djf 20151127 add for uboot logo:copy data to ram from nand */

        if (!((bmp->header.signature[0] == 'B') &&
                (bmp->header.signature[1] == 'M'))) {

#ifdef CONFIG_VIDEO_BMP_GZIP
            /*
              * Could be a gzipped bmp image, try to decrompress...
              */
            len = CONFIG_SYS_VIDEO_LOGO_MAX_SIZE;
            dst = malloc(CONFIG_SYS_VIDEO_LOGO_MAX_SIZE);
            if (dst == NULL) {
                printf("Error: malloc in gunzip failed!\n");
                return 1;
            }
            if (gunzip(dst, CONFIG_SYS_VIDEO_LOGO_MAX_SIZE,
                    (uchar *) bmp_image,
                    &len) != 0) {
                printf("Error: no valid bmp or bmp.gz image at %lx\n",
                        bmp_image);
                free(dst);
                return 1;
            }
            if (len == CONFIG_SYS_VIDEO_LOGO_MAX_SIZE) {
                printf("Image could be truncated "
                    "(increase CONFIG_SYS_VIDEO_LOGO_MAX_SIZE)!\n");
            }

            /*
              * Set addr to decompressed image
```

```
 */
bmp = (bmp_image_t *) dst;

if (!((bmp->header.signature[0] == 'B') &&
        (bmp->header.signature[1] == 'M'))) {
    printf("Error: no valid bmp.gz image at %lx\n",
            bmp_image);
    free(dst);
    return 1;
}
#else
    /* djf 20151127 add for uboot logo:copy data to ram from mmc */

    sprintf(buf, "mmc rescan");
    setenv("tempenv", buf);
    argv[0] = "run";
    argv[1] = "tempenv";
    argv[2] = NULL;

    if (do_run(NULL, 0, 2, argv))
    {
        printf("There is no valid bmp image at %lx\n and no mmc", bmp_image);
        return 1;
    }
    setenv("tempenv", NULL);

    sprintf(buf, "mw.b 0x82000000 0xff 0x200000");
    setenv("tempenv", buf);
    argv[0] = "run";
    argv[1] = "tempenv";
    argv[2] = NULL;

    if (do_run(NULL, 0, 2, argv))
    {
        printf("Error: mw.b err \n");
        return 1;
    }
    setenv("tempenv", NULL);


    sprintf(buf, "fatload mmc 0 0x82000000 logo.bmp");
    setenv("tempenv", buf);
    argv[0] = "run";
    argv[1] = "tempenv";
```

```
            argv[2] = NULL;

            if(do_run(NULL, 0, 2, argv))
            {
                printf("Error: fatload logo.bmp err \n");
                return 1;
            }
            setenv("tempenv", NULL);
            bmp =    (bmp_image_t *)0x82000000;

            if (!((bmp->header.signature[0] == 'B') &&
               (bmp->header.signature[1] == 'M')))
               {
                  printf("There is no valid bmp image at %lx\n or mmc", bmp_image);
                  return 1;
            }

            /* djf 20151127 add for uboot logo:copy data to ram from mmc */

/* //djf 20151127 del
            printf("Error: no valid bmp image at %lx\n or mmc", bmp_image);
            return 1;
*/
#endif /* CONFIG_VIDEO_BMP_GZIP */
      }

//    printf( "valid bmp image at %lx\n", bmp_image); //djf 20151126 add for test

      width = le32_to_cpu(bmp->header.width);
      height = le32_to_cpu(bmp->header.height);
      bpp = le16_to_cpu(bmp->header.bit_count);
      colors = le32_to_cpu(bmp->header.colors_used);
      compression = le32_to_cpu(bmp->header.compression);

      debug("Display-bmp: %ld x %ld    with %d colors\n",
            width, height, colors);

      if (compression != BMP_BI_RGB
#ifdef CONFIG_VIDEO_BMP_RLE8
            && compression != BMP_BI_RLE8
#endif
            ) {
            printf("Error: compression type %ld not supported\n",
                  compression);
```

```
#ifdef CONFIG_VIDEO_BMP_GZIP
        if (dst)
            free(dst);
#endif
        return 1;
    }

    padded_line = (((width * bpp + 7) / 8) + 3) & ~0x3;

//此宏定义必须打开
#ifdef CONFIG_SPLASH_SCREEN_ALIGN
    if (x == BMP_ALIGN_CENTER)
        x = max(0, (VIDEO_VISIBLE_COLS - width) / 2);
    else if (x < 0)
        x = max(0, VIDEO_VISIBLE_COLS - width + x + 1);

    if (y == BMP_ALIGN_CENTER)
        y = max(0, (VIDEO_VISIBLE_ROWS - height) / 2);
    else if (y < 0)
        y = max(0, VIDEO_VISIBLE_ROWS - height + y + 1);
#endif /* CONFIG_SPLASH_SCREEN_ALIGN */

    /*
     * Just ignore elements which are completely beyond screen
     * dimensions.
     */
    if ((x >= VIDEO_VISIBLE_COLS) || (y >= VIDEO_VISIBLE_ROWS))
        return 0;

    if ((x + width) > VIDEO_VISIBLE_COLS)
        width = VIDEO_VISIBLE_COLS - x;
    if ((y + height) > VIDEO_VISIBLE_ROWS)
        height = VIDEO_VISIBLE_ROWS - y;

    bmap = (uchar *) bmp + le32_to_cpu(bmp->header.data_offset);
    fb = (uchar *) (video_fb_address +
            ((y + height - 1) * VIDEO_COLS * VIDEO_PIXEL_SIZE) +
            x * VIDEO_PIXEL_SIZE);

#ifdef CONFIG_VIDEO_BMP_RLE8
    if (compression == BMP_BI_RLE8) {
        return display_rle8_bitmap(bmp, x, y, width, height);
    }
#endif
```

```
        /* We handle only 4, 8, or 24 bpp bitmaps */
        switch (le16_to_cpu(bmp->header.bit_count)) {
        case 4:
                padded_line -= width / 2;
                ycount = height;

                switch (VIDEO_DATA_FORMAT) {
                case GDF_32BIT_X888RGB:
                        while (ycount--) {
                                WATCHDOG_RESET();
                                /*
                                 * Don't assume that 'width' is an
                                 * even number
                                 */
                                for (xcount = 0; xcount < width; xcount++) {
                                        uchar idx;

                                        if (xcount & 1) {
                                                idx = *bmap & 0xF;
                                                bmap++;
                                        } else
                                                idx = *bmap >> 4;
                                        cte = bmp->color_table[idx];
                                        FILL_32BIT_X888RGB(cte.red, cte.green,
                                                           cte.blue);
                                }
                                bmap += padded_line;
                                fb -= (VIDEO_VISIBLE_COLS + width) *
                                        VIDEO_PIXEL_SIZE;
                        }
                        break;
                default:
                        puts("4bpp bitmap unsupported with current "
                             "video mode\n");
                        break;
                }
                break;

        case 8:
//              printf( "the bmp is 8bpp\n" ); //djf 20151126 add for test
                padded_line -= width;
                if (VIDEO_DATA_FORMAT == GDF__8BIT_INDEX) {
                        /* Copy colormap */
```

```
                for (xcount = 0; xcount < colors; ++xcount) {
                        cte = bmp->color_table[xcount];
                        video_set_lut(xcount, cte.red, cte.green,
                                        cte.blue);
                }
        }
        ycount = height;
        switch (VIDEO_DATA_FORMAT) {
        case GDF__8BIT_INDEX:
                while (ycount--) {
                        WATCHDOG_RESET();
                        xcount = width;
                        while (xcount--) {
                                *fb++ = *bmap++;
                        }
                        bmap += padded_line;
                        fb -= (VIDEO_VISIBLE_COLS + width) *
                                        VIDEO_PIXEL_SIZE;
                }
                break;
        case GDF__8BIT_332RGB:
                while (ycount--) {
                        WATCHDOG_RESET();
                        xcount = width;
                        while (xcount--) {
                                cte = bmp->color_table[*bmap++];
                                FILL_8BIT_332RGB(cte.red, cte.green,
                                        cte.blue);
                        }
                        bmap += padded_line;
                        fb -= (VIDEO_VISIBLE_COLS + width) *
                                        VIDEO_PIXEL_SIZE;
                }
                break;
        case GDF_15BIT_555RGB:
                while (ycount--) {
#if defined(VIDEO_FB_16BPP_PIXEL_SWAP)
                        int xpos = x;
#endif
                        WATCHDOG_RESET();
                        xcount = width;
                        while (xcount--) {
                                cte = bmp->color_table[*bmap++];
#if defined(VIDEO_FB_16BPP_PIXEL_SWAP)
```

```
                        fill_555rgb_pswap(fb, xpos++, cte.red,
                                        cte.green,
                                        cte.blue);
                        fb += 2;
#else
                        FILL_15BIT_555RGB(cte.red, cte.green,
                                        cte.blue);
#endif
                    }
                    bmap += padded_line;
                    fb -= (VIDEO_VISIBLE_COLS + width) *
                                    VIDEO_PIXEL_SIZE;
                }
                break;
            case GDF_16BIT_565RGB:
                while (ycount--) {
                    WATCHDOG_RESET();
                    xcount = width;
                    while (xcount--) {
                        cte = bmp->color_table[*bmap++];
                        FILL_16BIT_565RGB(cte.red, cte.green,
                                        cte.blue);
                    }
                    bmap += padded_line;
                    fb -= (VIDEO_VISIBLE_COLS + width) *
                                    VIDEO_PIXEL_SIZE;
                }
                break;
            case GDF_32BIT_X888RGB:
//                  printf( "The LCD bit type is 32bit\n" ); //djf 20151126 add for test
                while (ycount--) {
                    WATCHDOG_RESET();
                    xcount = width;
                    while (xcount--) {
                        cte = bmp->color_table[*bmap++];
                        FILL_32BIT_X888RGB(cte.red, cte.green,
                                        cte.blue);
                    }
                    bmap += padded_line;
                    fb -= (VIDEO_VISIBLE_COLS + width) *
                                    VIDEO_PIXEL_SIZE;
                }
                break;
            case GDF_24BIT_888RGB:
```

```
                while (ycount--) {
                        WATCHDOG_RESET();
                        xcount = width;
                        while (xcount--) {
                                cte = bmp->color_table[*bmap++];
                                FILL_24BIT_888RGB(cte.red, cte.green,
                                                cte.blue);
                        }
                        bmap += padded_line;
                        fb -= (VIDEO_VISIBLE_COLS + width) *
                                        VIDEO_PIXEL_SIZE;
                }
                break;
        }
        break;
    case 24:
        padded_line -= 3 * width;
        ycount = height;
        switch (VIDEO_DATA_FORMAT) {
        case GDF__8BIT_332RGB:
                while (ycount--) {
                        WATCHDOG_RESET();
                        xcount = width;
                        while (xcount--) {
                                FILL_8BIT_332RGB(bmap[2], bmap[1],
                                                bmap[0]);
                                bmap += 3;
                        }
                        bmap += padded_line;
                        fb -= (VIDEO_VISIBLE_COLS + width) *
                                        VIDEO_PIXEL_SIZE;
                }
                break;
        case GDF_15BIT_555RGB:
                while (ycount--) {
#if defined(VIDEO_FB_16BPP_PIXEL_SWAP)
                        int xpos = x;
#endif
                        WATCHDOG_RESET();
                        xcount = width;
                        while (xcount--) {
#if defined(VIDEO_FB_16BPP_PIXEL_SWAP)
                                fill_555rgb_pswap(fb, xpos++, bmap[2],
                                                bmap[1], bmap[0]);
```

```
                        fb += 2;
#else
                        FILL_15BIT_555RGB(bmap[2], bmap[1],
                                bmap[0]);
#endif
                        bmap += 3;
                    }
                    bmap += padded_line;
                    fb -= (VIDEO_VISIBLE_COLS + width) *
                            VIDEO_PIXEL_SIZE;
                }
                break;
            case GDF_16BIT_565RGB:
                while (ycount--) {
                    WATCHDOG_RESET();
                    xcount = width;
                    while (xcount--) {
                        FILL_16BIT_565RGB(bmap[2], bmap[1],
                                bmap[0]);
                        bmap += 3;
                    }
                    bmap += padded_line;
                    fb -= (VIDEO_VISIBLE_COLS + width) *
                            VIDEO_PIXEL_SIZE;
                }
                break;
            case GDF_32BIT_X888RGB:
                while (ycount--) {
                    WATCHDOG_RESET();
                    xcount = width;
                    while (xcount--) {
                        FILL_32BIT_X888RGB(bmap[2], bmap[1],
                                bmap[0]);
                        bmap += 3;
                    }
                    bmap += padded_line;
                    fb -= (VIDEO_VISIBLE_COLS + width) *
                            VIDEO_PIXEL_SIZE;
                }
                break;
            case GDF_24BIT_888RGB:
                while (ycount--) {
                    WATCHDOG_RESET();
                    xcount = width;
```

```
                        while (xcount--) {
                            FILL_24BIT_888RGB(bmap[2], bmap[1],
                                    bmap[0]);
                            bmap += 3;
                        }
                        bmap += padded_line;
                        fb -= (VIDEO_VISIBLE_COLS + width) *
                                VIDEO_PIXEL_SIZE;
                    }
                    break;
                default:
                    printf("Error: 24 bits/pixel bitmap incompatible "
                        "with current video mode\n");
                    break;
                }
                break;
            default:
                printf("Error: %d bit/pixel bitmaps not supported by U-Boot\n",
                    le16_to_cpu(bmp->header.bit_count));
                break;
            }

        #ifdef CONFIG_VIDEO_BMP_GZIP
            if (dst) {
                free(dst);
            }
        #endif

            if (cfb_do_flush_cache)
                flush_cache(VIDEO_FB_ADRS, VIDEO_SIZE);
            return (0);
        }
```

五、问题

1.uboot 中 logo 和 kernel 中 logo 因色深度影响，不完全一致

2.uboot 显示 logo 会有轻微波纹

3.修改 nand 的分区等时，需要修改最后一步中自己添加的拷贝 bmp 图片信息到内存相关的代码，涉及到的地址与分区及为图片分配空间大小有关，现在的分区是 MLO，MLO 三个备用分区，每个分区 2M，其中第三个备用分区（0x600000-0x7fffff）用来保存 logo 图片，uboot 分区。接下来是 2M 的 Uboot，2M 的 Env，8M 的 kernel，剩下的 rootfs。

4.记得添加 LCD 引脚配置：/board/ti/am335x/mux.c->enable_board_pin_mux():

```
void enable_board_pin_mux(struct am335x_baseboard_id *header)
{
```

```
        /* Do board-specific muxes. */
        if (board_is_bone(header)) {
                /* Beaglebone pinmux */
                configure_module_pin_mux(i2c1_pin_mux);
                configure_module_pin_mux(mii1_pin_mux);
                configure_module_pin_mux(mmc0_pin_mux);
#ifndef CONFIG_NOR
                configure_module_pin_mux(mmc1_pin_mux);
#endif
#if defined(CONFIG_NOR) && !defined(CONFIG_NOR_BOOT)
                configure_module_pin_mux(bone_norcape_pin_mux);
#endif
        } else if (board_is_gp_evm(header)) {
                /* General Purpose EVM */
                //unsigned short profile = detect_daughter_board_profile();
                configure_module_pin_mux(rgmii1_pin_mux);
                configure_module_pin_mux(mmc0_pin_mux);
                /* In profile #2 i2c1 and spi0 conflict. */
/*dragoniye */
                configure_module_pin_mux(nand_pin_mux);
                //configure_module_pin_mux(spi0_pin_mux);//djf 20150105 del pin not
use
                //configure_module_pin_mux(i2c1_pin_mux);//djf 20150105 del pin not
use
                configure_module_pin_mux( lcd_pin_mux);
                //configure_module_pin_mux( gpio_pin_mux);
                ……
        }
}
```

其中 lcd_pin_mux[]如下：

```
/* djf 20150105 add start */

static struct module_pin_mux lcd_pin_mux[] = {
        {OFFSET(gpmc_ad8), (MODE(1))},/* LCD_DATA23 */
        {OFFSET(gpmc_ad9), (MODE(1))},
        {OFFSET(gpmc_ad10), (MODE(1))},
        {OFFSET(gpmc_ad11), (MODE(1))},
        {OFFSET(gpmc_ad12), (MODE(1))},
        {OFFSET(gpmc_ad13), (MODE(1))},
        {OFFSET(gpmc_ad14), (MODE(1))},
        {OFFSET(gpmc_ad15), (MODE(1))},
        {OFFSET(lcd_data0), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data1), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data2), (MODE(0) | PULLUDDIS)},
```

```
        {OFFSET(lcd_data3), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data4), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data5), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data6), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data7), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data8), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data9), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data10), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data11), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data12), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data13), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data14), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_data15), (MODE(0) | PULLUDDIS)},
        {OFFSET(lcd_vsync), (MODE(0))},
        {OFFSET(lcd_hsync), (MODE(0))},
        {OFFSET(lcd_pclk), (MODE(0))},
        {OFFSET(lcd_ac_bias_en), (MODE(0))},
        {-1},
    };

    /* djf 20150105 add end */
```

5.LCD 背光打开是在/arch/arm/lib/board.c->board_init_r()->
board_late_init()(/board/ti/am335x/board.c)-> tps65217_open_backlight()中完成的，但是默认的是没有打开，需修改如下：

```
    void tps65217_open_backlight(void)
    {
        printf("TPS65217 open backlight entry \n");
        if (i2c_probe(TPS65217_CHIP_PM))
                return;
        printf("TPS65217 probe OK \n");

        if (tps65217_reg_write(TPS65217_PROT_LEVEL_NONE,
                               TPS65217_WLEDCTRL2,
                               40,
                               TPS65217_MASK_ALL_BITS))
            printf("TPS65217 set wled duty error \n");

        if (tps65217_reg_write(TPS65217_PROT_LEVEL_NONE,
                               TPS65217_WLEDCTRL1,
                               8,
                               TPS65217_WLEDCTRL1_ON_OFF_BITMASK))// djf 2015 1126
    change value 1 to 8
            printf("TPS65217 set wled On error \n");
        printf("TPS65217 open backlight OK \n");
```

```
}
```