1 软件架构

1.1 驱动软件架构

1）由于tvout有几大模块组成(vp,mixer,cec,hpd)，同时这些模块都集成在AP这边。所以驱动实现为一个平台驱动（platform_driver）。

```
ret = platform_driver_register(&s5p_tvout_driver);

static struct platform_driver s5p_tvout_driver = {
        .probe          = s5p_tvout_probe,
        .remove         = s5p_tvout_remove,
        .driver         = {
        .name           = "s5p-tvout",
        .owner          = THIS_MODULE,
        .pm     = &s5p_tvout_pm_ops
        },
};
```

在驱动的probe函数中，对几大模块进行初始化,同时初始化时钟。

```
if (s5p_tvout_clk_get(pdev, &s5ptv_status) < 0)
        goto err;

        if (s5p_vp_ctrl_constructor(pdev) < 0)
        goto err;

        /* s5p_mixer_ctrl_constructor must be called
        before s5p_tvif_ctrl_constructor */
        if (s5p_mixer_ctrl_constructor(pdev) < 0)
        goto err;

        if (s5p_tvif_ctrl_constructor(pdev) < 0)
        goto err;

        if (s5p_tvout_v4l2_constructor(pdev) < 0)
        goto err;
```

以上四个 constructor 的结构基本类似。

```
int s5p_vp_ctrl_constructor(struct platform_device *pdev)
{
        int ret = 0;

        ret = s5p_tvout_map_resource_mem(pdev,
                s5p_vp_ctrl_private.reg_mem.name,
                &(s5p_vp_ctrl_private.reg_mem.base),
                &(s5p_vp_ctrl_private.reg_mem.res));

        s5p_vp_ctrl_private.clk.ptr =
                clk_get(&pdev->dev, s5p_vp_ctrl_private.clk.name);


        s5p_vp_init(s5p_vp_ctrl_private.reg_mem.base);
        s5p_vp_ctrl_init_private();

        return 0;
```

四个模块构建的 constructor 过程如下：

1)从各自的平台设备(platform_device)中得到 IO 内存和中断等资源。

2)对 IO 内存资源进行映射(io_remap),对中断进行申请(request_irq)。

3)获取时钟(clk_get)。

4)私有数据的初始化(init_private)。

四个模块构建 constructor 的功能如下：

1) s5p_vp_ctrl_constructor(pdev) vp(video processor 的初始化)

2) s5p_mixer_ctrl_constructor(pdev)(mixer 的初始化)

3) s5p_tvif_ctrl_constructor(pdev) (sdo 和 hdmi 的初始化)

4) s5p_tvout_v4l2_constructor (video device 的注册) 驱动向外提供 v4l2 (video for linux 2)接口

最后申请framebuffer

```
	/* prepare memory */
		if (s5p_tvout_fb_alloc_framebuffer(&pdev->dev))
		goto err;

		if (s5p_tvout_fb_register_framebuffer(&pdev->dev))
		goto err;
```

2）tvout所用到的资源 ,在文件dev-tvout.c中,这些资源主要包括IO内存和IRQ资源，资源包括了起始地址和终止地址。这个文件中也提供了相应的平台设备(platform_device)结构。这些平台设备变量通过EXPORT_SYMBOL()导出。在mach初始化（mach-m040.c）的时候被设置。

```
static struct resource s5p_tvout_resources[] = {
        [0] = {
        .start  = S5P_PA_TVENC,
        .end    = S5P_PA_TVENC + S5P_SZ_TVENC - 1,
        .flags  = IORESOURCE_MEM,
        .name        = "s5p-sdo"
        },
        [1] = {
        .start  = S5P_PA_VP,
        .end    = S5P_PA_VP + S5P_SZ_VP - 1,
        .flags  = IORESOURCE_MEM,
        .name        = "s5p-vp" //video processor
        },
        [2] = {
        .start  = S5P_PA_MIXER,
        .end    = S5P_PA_MIXER + S5P_SZ_MIXER - 1,
        .flags  = IORESOURCE_MEM,
        .name        = "s5p-mixer"
        },
        [3] = {
        .start  = S5P_PA_HDMI,
        .end    = S5P_PA_HDMI + S5P_SZ_HDMI - 1,
        .flags  = IORESOURCE_MEM,
        .name        = "s5p-hdmi"
        },
        [4] = {
        .start  = S5P_I2C_HDMI_PHY,
        .end    = S5P_I2C_HDMI_PHY + S5P_I2C_HDMI_SZ_PHY - 1,
        .flags  = IORESOURCE_MEM,
        .name        = "s5p-i2c-hdmi-phy"
        },
        [5] = {
        .start  = IRQ_MIXER,
        .end    = IRQ_MIXER,
        .flags  = IORESOURCE_IRQ,
        .name        = "s5p-mixer"
        },
        [6] = {
        .start  = IRQ_HDMI,
        .end    = IRQ_HDMI,
        .flags  = IORESOURCE_IRQ,
        .name        = "s5p-hdmi"
        },
        [7] = {
        .start  = IRQ_TVENC,
        .end    = IRQ_TVENC,
        .flags  = IORESOURCE_IRQ,
        .name        = "s5p-sdo"
        },
};
```

1.2 Android 端软件架构

由于驱动向外提供了v4l2(video for linux 2)接口，所以上层得以控制设备。

1.1 ioctl 接口 V4L2 OUTPUT API Lists

IOCTL Name	Descriptions
VIDIOC_QUERYCAP	Query device capabilities
VIDIOC_ENUMSTD	Enumerate supported video standards
VIDIOC_G_STD	Query the video standard of the current input
VIDIOC_S_STD	Select the video standard of the current input
VIDIOC_ENUMOUTPUT	Enumerate video outputs
VIDIOC_G_OUTPUT	Query the current video output
VIDIOC_S_OUTPUT	Select the current video output
VIDOC_G_CTRL	Get the value of a control
VIDIOC_S_CTRL	Set the value of a conrol

1.2 ioctl OVERLAY API Lists

IOCTL Name	Descriptions
VIDIOC_ENUM_FMT	Enumerate image formats
VIDIOC_G_FMT	Get the data format
VIDIOC_S_FMT	Set the data format
VIDIOC_CROPCAP	Information about the video cropping and scaling abilities
VIDIOC_G_CROP	Get the current cropping rectangle
VIDIOC_S_CROP	Set the current cropping rectangle
VIDIOC_G_FBUF	Get frame buffer overlay parameters
VIDIOC_S_FBUF	Set frame buffer overlay parameters
VIDIOC_OVERLAY	Start or stop video overlay

1.3 FrameBuffer API Lists

IOCTL Name	IOCTL Code	Descriptions
S5PTVFB_WIN_POSITION	_IOW ('F', 213, struct s5ptvfb_user_window)	Configures the offset to display in the TV

S5PTVFB_WIN_SET_PLANE_ALPHA	_IOW ('F', 214, struct s5ptvfb_user_plane_alpha)	Configures plane alpha blending
S5PTVFB_WIN_SET_CHROMA	_IOW ('F', 215, struct s5ptvfb_user_chroma)	Configures chroma key information
S5PTVFB_SCALING	_IOW ('F', 222, struct s5ptvfb_user_scaling)	Configure horizontal, vertical scaling value

有了这些接口之后，上层只需要对设备文件/dev/video14(VIDEO_OUTPUT), /dev/video21(VIDEO_OVERLAY)进行 ioctl 函数调用，就可以控制设备了。