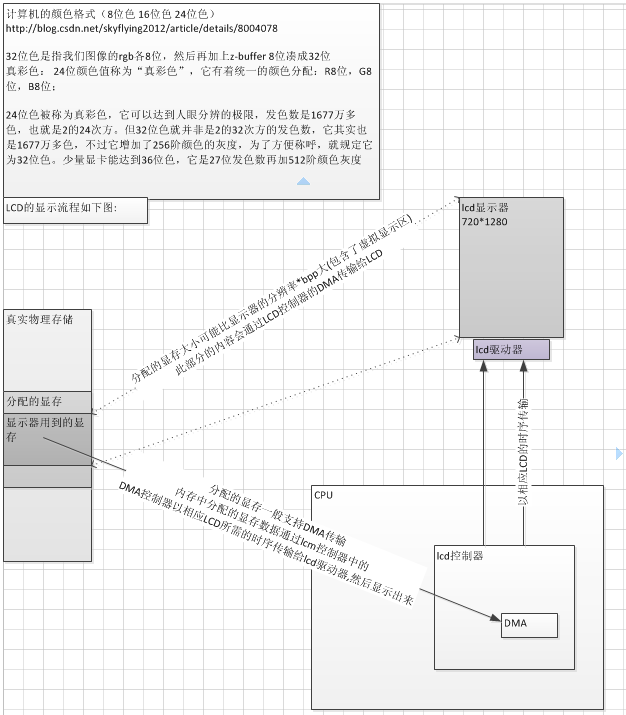
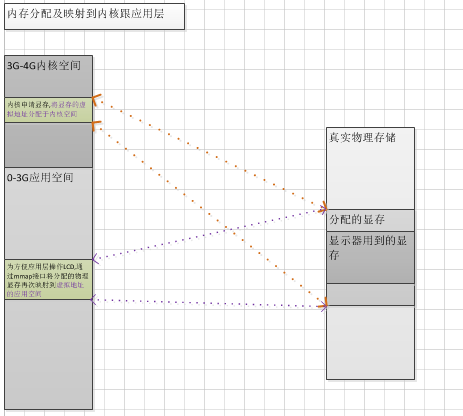
**姓名：王超群**

**日期：2016-04-10**

**内容：linxu内核framerbuffer框架图**

**其他：**





代码实现介绍:

linux中实现LCD控制主要使用framebuffer框架:

主要是struct fb\_info 结构体,每一个用framebuffer框架的显示设备都必须对应一个struct fb\_info结构体

此结构体中有如下元素:

struct fb\_info {

struct fb\_var\_screeninfo var;//LCD可变参数

struct fb\_fix\_srceeninfo fix;//LCD固定参数

struct fb\_videomode \*mode;//目前video模式

struct fb\_ops \*fbops;

unsigned long screen\_size;//需要分配的显存大小

char \_\_iomem \*screen\_base;//显存的虚拟地址(起始地址)

void \*par;

...

}

**驱动代码使用framebuffer框架需完成如下几步:**

1:申请fb\_info结构体,初始化此结构体,如填充var,fix,

2:,根据LCD的具体时序,初始化LCD控制器,发出发出合适的时序

3:内核申请显存(考虑cache一致性问题,需用DMA方式申请),

并将映射到内核的虚拟地址给fb\_info.screen\_base,大小给fix.smem\_len = len;物理地址给fix.smem\_len = \_\_pa(fb\_info.screen\_base);**DMA不经过PMU所需的是物理地址**

4:注册帧缓冲设备register\_framebuffer(struct fb\_info\*);

**应用层使用帧缓冲设备是,按如下步骤:**

1:打开帧缓冲设备

**int fbdev = open(“/dev/fb0”, O\_RDWR);**

2:获取屏幕的分辨率,计算缓冲区的大小

struct fb\_var\_screeninfo vifo;

**ioctl**(fbdev,**FBIOGET\_VSCREENINFO,&vifo**);

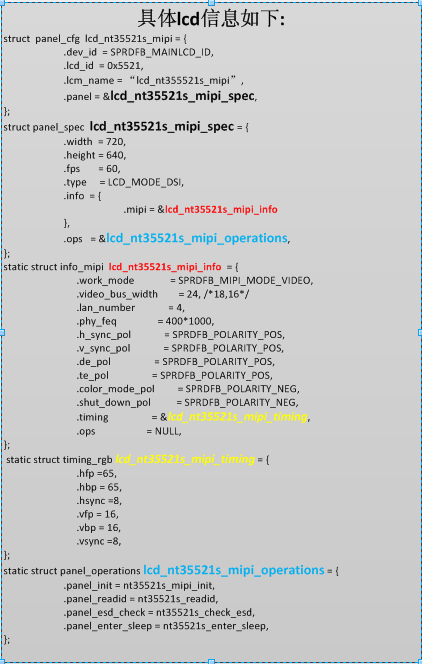
**unsigned long screensize = vinfo.xres \* vinfo.yres;**

3:调用mmap函数将显存映射到应用空间

**char \* fbp = (char \*)mmap(0,screensize, PROT\_READ|PROT\_WRITE,MAP\_SHARED);**

4:调用返回的应用空间显存地址来操作显存

**\*((unsigned short\*)(fbp + location)) = rgb;**//直接将fbp为开始地址的偏移为location内存存放对应RGB值



展讯代码分析:

static int sprdfb\_probe(struct platform\_device \*pdev)

{

struct fb\_info \*fb = NULL;

struct fprdfb\_device \*dev = NULL;

fb = framebuffer\_alloc(sizeof (struct sprdfb\_device),&pdev->dev);//分配的空间大小为**fb\_info结构体大小+sprdfb\_device结构大小和**

dev = fb->par;

dev->fb = fb;

#if CONFIG\_OF

获取platform\_device中的信息

#else

获取platform\_device中的信息

#endif

switch (SPRD\_IN\_DATA\_TYPE){

case SPRD\_IN\_DATA\_TYPE\_ABGR888:

dev->bpp = 32;

break;

case SPRD\_IN\_DATA\_TYPE\_BGR565:

dev->bpp = 16;

break;

default:

dev->bpp = 32;

break;

}

dev->ctrl = &sprdfb\_dispc\_ctrl;

if(sprdfb\_panel\_get(dev)){//获取当前硬件LCD信息,根据LCD的id来一个一个匹配,返回 dev->panel = **lcd\_nt35521s\_mipi\_spec;**

dev->panel\_ready = true;

}

//如下是展开 setup\_fb\_mem(struct sprdfb\_device \*dev,struct platform\_device \*pdev);

{

uint32\_t len = dev->panel->width \* dev->panel->height \* (dev->bpp/8) \* FRAMEBUFFER\_NR;//**分配FRAMEBUFFER\_NR个LCD分辨率大小的显存**

void \*addr = (void \*)\_\_get\_free\_pages(GFP\_ATOMIC | \_\_GFP\_ZERO,get\_order(len));

dev->fb->fix.smem\_start = \_\_pa(addr);//将分配的虚拟地址对应的物理地址给smem\_start,主要是因为**如果用DMA传输,DMA不经过PMU所需的是物理地址**

dev->fb->fix.smem\_len = len;

dev->fb->screen\_base = (char \*) addr;

}

//如下是展开setup\_fb\_info(struct sprdfb\_device \*dev)

//------------------------------------------------------------------------------------------------------------------------------start

{

struct fb\_info \*fb = dev->fb;

struct panel\_spec \*panel = dev->panle;//具体lcm结构体

fb->fbops = &sprdfb\_ops;

strncpy(fb->fix.id,”sprdfb”,16);

fb->fix.line\_length = panel->width \* dev->bpp / 8; //1行的字节数

fb->var.xres = panel->width;//x可见解析度

fb->var.yres = panel->height;//y可见解析度

fb->var.width = panel->width;//宽度

fb->var.height = panel->height;//高度

fb->var.xres\_virtual = panel->width;//x虚拟解析度

fb->var.yres\_virtual = panel->height \* FRAMEBUFFER\_NR;//Y虚拟解析度

fb->var.bits\_per\_pixel = dev->bpp;

}

//以下为比较重要的像素时钟 pixclock !!!,从具体LCD结构体设备中获取如上fps = 60

if(0 != dev->panel->fps){

fb->var.pixclock = ((1000000000 /panel->width) \* 1000) / (dev->panel->fps \* panel->height);

}else{

fb->var.pixclock = ((1000000000 /panel->width) \* 1000) / (60 \* panel->height)

}

fb->var.yoffset = 0;//虚拟到可见之间的偏移

//设置RGB 位域

if(dev->bpp == 32){

fb->var.red.offset = 16;//位域偏移

fb->var.red.length = 8;//位域长度

fb->var.red.msb\_right = 0;//!0 MSB在右边

fb->var.green.offset = 8;

fb->var.green.length = 8;

fb->var.green.msb\_right = 0;

fb->var.blue.offset = 0;

fb->var.blue.length = 8;

fb->var.blue.msb\_right = 0;

}

//分配调色板

int r = fb\_alloc\_cmap(&fb->cmap,16,0);

static unsigned PP[16] ;

fb->pseudo\_palette = PP;//伪16色颜色表

PP[0] = 0;

for(r = 1;r<16,r++){

PP[r] = 0xffffffff;//32bit

}

//------------------------------------------------------------------------------------------------------------------------------end

//注册fb\_info

register\_framebuffer(fb);

platform\_set\_drvdata(pdev,dev);

sprdfb\_create\_sysfs(dev);

dev->ctrl->init(dev);**//设置LCD控制器**

#if CONFIG\_HAS\_EARLYSUSPEND

dev->early\_suspend.suspend = sprdfb\_early\_suspend;

dev->early\_suspend.resume = sprdfb\_late\_resume;

dev->early\_suspend.level = EARLY\_SUSPEND\_LEVEL\_DISABLE\_FB;

register\_early\_suspend(&dev->early\_suspend);  
#endif

//prob结束

}