努力成为 linux kernel hacker 的人李万鹏原创作品，为梦而战。转载请标明出处

http://blog.csdn.net/woshixingaaa/archive/2011/05/17/6426203.aspx

内核启动时，会调用 s3c24xx_register_clock 函数注册很多时钟，所谓注册，就是在一个链表中保存各种"struct clk*"结构指针，这些"struct clk"结构有:clk_f(表示 FCLK)，clk_h(表示 HCLK)，clk_p(表示 PCLK)等。然后可以通过 clk_get_rate 函数获得获得某类时钟频率。下面到内核中分析一下源码，这里跟踪内核启动时 clock system 的初始化过程：

```
1. asmlinkage void __init start_kernel(void)
2. {
3.     .............
4.     setup_arch(&command_line);
5.     .............
6. }
```

start_kernel 调用了 setup_arch(&command_line)：

```
1. void __init setup_arch(char **cmdline_p)
2. {
3.     ..........
4.     paging_init(mdesc);
5.     ..........
6. }
```

setup_arch 调用了 paging_init(mdesc)：

```
1. void __init paging_init(struct machine_desc *mdesc){
2.     .............
3.     devicemaps_init(mdesc);
4.     .............
5. }
```

paging_init 调用了 devicemaps_init(mdesc)：

```
1. static void __init devicemaps_init(struct machine_desc *mdesc)
2. {
3.     ..................
4.     /*这里调用具体体系结构的 map_io 函数*/
5.     if (mdesc->map_io)
6.         mdesc->map_io();
7.     ..................
8. }
```

在我们板子的文件中查到这个 map_io 函数：

```
1. static void __init smdk2440_map_io(void)
2. {
3.     s3c24xx_init_io(smdk2440_iodesc, ARRAY_SIZE(smdk2440_iodesc));
4.     s3c24xx_init_clocks(12000000);
5.     s3c24xx_init_uarts(smdk2440_uartcfgs, ARRAY_SIZE(smdk2440_uartcfgs));
6. }
```

这个函数有一句 s3c24xx_init_clocks(12000000);跟踪进去：

```
1. /* s3c24xx_init_clocks
2. *
3. * Initialise the clock subsystem and associated information from the
4. * given master crystal value.
5. *
6. * xtal  = 0 -> use default PLL crystal value (normally 12MHz)
7. *      != 0 -> PLL crystal value in Hz
8. */
9.
10.void __init s3c24xx_init_clocks(int xtal)
11.{
12.    if (xtal == 0)
13.        xtal = 12*1000*1000;
14.
15.    if (cpu == NULL)
16.        panic("s3c24xx_init_clocks: no cpu setup?/n");
17.
18.    if (cpu->init_clocks == NULL)
19.        panic("s3c24xx_init_clocks: cpu has no clock init/n");
20.    else
21.        (cpu->init_clocks)(xtal);
22.}
```

这个函数是设置晶振的频率为 12M，也就是我板子上的晶振 12M。注意这里最后一句：(cpu->init_clocks)(xtal);我们要查看 cpu_table 了。

```
1. static struct cpu_table cpu_ids[] __initdata = {
2.     ......................
3.     {
4.         .idcode     = 0x32440001,
5.         .idmask     = 0xffffffff,
6.         .map_io     = s3c244x_map_io,
7.         .init_clocks    = s3c244x_init_clocks,
8.         .init_uarts = s3c244x_init_uarts,
9.         .init       = s3c2440_init,
10.        .name       = name_s3c2440a
11.    },
12.    ......................
```

```
13.};
```

在 cpu_table 中可以找到这个 init_clocks 函数，也就是我们苦苦寻觅的 clock system 初始化函数了，真是众里寻他千百度，那人却在灯火阑珊处。

```
1.  void __init s3c244x_init_clocks(int xtal)
2.  {
3.     /* initialise the clocks here, to allow other things like the
4.      * console to use them, and to add new ones after the initialisation
5.      */
6.
7.     s3c24xx_register_baseclocks(xtal);
8.     s3c244x_setup_clocks();
9.     s3c2410_baseclk_add();
10.}
```

这个 s3c244x_init_clocks 完成了 clock system 全部的初始化工作。现在一个一个来分析里边的 3 个函数。s3c24xx_register_baseclocks()函数在 arch/arm/plat-s3c/clock.c 中实现如下：这里对基本的时钟 clk_xtal,clk_mpll,clk_upll,clk_f,clk_h,clk_p 进行了注册。

```
1.  int __init s3c24xx_register_baseclocks(unsigned long xtal)
2.  {
3.     printk(KERN_INFO "S3C24XX Clocks, (c) 2004 Simtec Electronics/n");
4.     clk_xtal.rate = xtal;
5.     /* register our clocks */
6.
7.     if (s3c24xx_register_clock(&clk_xtal) < 0)
8.        printk(KERN_ERR "failed to register master xtal/n");
9.
10.    if (s3c24xx_register_clock(&clk_mpll) < 0)
11.       printk(KERN_ERR "failed to register mpll clock/n");
12.
13.    if (s3c24xx_register_clock(&clk_upll) < 0)
14.       printk(KERN_ERR "failed to register upll clock/n");
15.
16.    if (s3c24xx_register_clock(&clk_f) < 0)
17.       printk(KERN_ERR "failed to register cpu fclk/n");
18.
19.    if (s3c24xx_register_clock(&clk_h) < 0)
20.       printk(KERN_ERR "failed to register cpu hclk/n");
21.
22.    if (s3c24xx_register_clock(&clk_p) < 0)
23.       printk(KERN_ERR "failed to register cpu pclk/n");
24.
25.    return 0;
```

```
26.}
```

下边看一下这个注册函数，主要任务就是把 struct clk 结构添加到 clocks 链表中。

```
1. /* clock information */
2. static LIST_HEAD(clocks);
```

这个是链表的头的注册函数。注册的 struct clk 结构体都要添加到这个 clocks 链表中。

```
1. /* initialise the clock system */
2.
3. int s3c24xx_register_clock(struct clk *clk)
4. {
5.     .........
6.     list_add(&clk->list, &clocks);
7.     .........
8. }
```

现在来看第二个函数：它的任务就是设置 fclk，hclk，pclk，相信如果认真写过 arm 裸机程序的人一定很容易看懂下边的代码了，可以对照 s3c2440 的手册来看的。

```
1. void __init_or_cpufreq s3c244x_setup_clocks(void)
2. {
3.     ..................
4.     s3c24xx_setup_clocks(fclk, hclk, pclk);
5. }
```

这里调用了一个 s3c24xx_setup_clocks 函数，下面看它的实现：

```
1. /* initalise all the clocks */
2. void __init_or_cpufreq s3c24xx_setup_clocks(unsigned long fclk,
3.                                             unsigned long hclk,
4.                                             unsigned long pclk)
5. {
6.     clk_upll.rate = s3c24xx_get_pll(__raw_readl(S3C2410_UPLLCON),
7.                                     clk_xtal.rate);
8.
9.     clk_mpll.rate = fclk;
10.    clk_h.rate = hclk;
11.    clk_p.rate = pclk;
12.    clk_f.rate = fclk;
13.}
```

就是把得到的 fclk,hclk,pclk 赋值相应结构体。

下面来看第三个函数，这个主要就是对外设的 struct clk 进行注册。这个函数一共分两部分，有两个数组，一个是 init_clocks，也就是在 boot 时需要提供时钟

的，一个是 init_clocks_disable，这里的每个成员都是在 boot 的时候需要 disable 时钟的。这两个数组分别进行注册，但是注册 init_clocks_disable 数组中成员的 for 循环中调用了 s3c2410_clkcon_enable(clkp, 0);也就是将相应的 clk disable 掉。

```
1.  int __init s3c2410_baseclk_add(void)
2.  {
3.     unsigned long clkslow = __raw_readl(S3C2410_CLKSLOW);
4.     unsigned long clkcon  = __raw_readl(S3C2410_CLKCON);
5.     struct clk *clkp;
6.     struct clk *xtal;
7.     int ret;
8.     int ptr;
9.
10.    clk_upll.enable = s3c2410_upll_enable;
11.
12.    if (s3c24xx_register_clock(&clk_usb_bus) < 0)
13.       printk(KERN_ERR "failed to register usb bus clock/n");
14.
15.    /* register clocks from clock array */
16.
17.    clkp = init_clocks;
18.    for (ptr = 0; ptr < ARRAY_SIZE(init_clocks); ptr++, clkp++) {
19.       /* ensure that we note the clock state */
20.
21.       clkp->usage = clkcon & clkp->ctrlbit ? 1 : 0;
22.
23.       ret = s3c24xx_register_clock(clkp);
24.       if (ret < 0) {
25.          printk(KERN_ERR "Failed to register clock %s (%d)/n",
26.                clkp->name, ret);
27.       }
28.    }
29.
30.    /* We must be careful disabling the clocks we are not intending to
31.     * be using at boot time, as subsystems such as the LCD which do
32.     * their own DMA requests to the bus can cause the system to lockup
33.     * if they where in the middle of requesting bus access.
34.     *
35.     * Disabling the LCD clock if the LCD is active is very dangerous,
36.     * and therefore the bootloader should be careful to not enable
37.     * the LCD clock if it is not needed.
38.     */
39.
```

```
40.  /* install (and disable) the clocks we do not need immediately */
41.
42.  clkp = init_clocks_disable;
43.  for (ptr = 0; ptr < ARRAY_SIZE(init_clocks_disable); ptr++, clkp++) {
44.
45.      ret = s3c24xx_register_clock(clkp);
46.      if (ret < 0) {
47.          printk(KERN_ERR "Failed to register clock %s (%d)/n",
48.                 clkp->name, ret);
49.      }
50.
51.      s3c2410_clkcon_enable(clkp, 0);
52.  }
53.
54.  /* show the clock-slow value */
55.
56.  xtal = clk_get(NULL, "xtal");
57.
58.  printk("CLOCK: Slow mode (%ld.%ld MHz), %s, MPLL %s, UPLL %s/n",
59.         print_mhz(clk_get_rate(xtal) /
60.           ( 2 * S3C2410_CLKSLOW_GET_SLOWVAL(clkslow))),
61.         (clkslow & S3C2410_CLKSLOW_SLOW) ? "slow" : "fast",
62.         (clkslow & S3C2410_CLKSLOW_MPLL_OFF) ? "off" : "on",
63.         (clkslow & S3C2410_CLKSLOW_UCLK_OFF) ? "off" : "on");
64.
65.  s3c_pwmclk_init();
66.  return 0;
67.}
```

在 arch/arm/plat-s3c/clock.c 中实现了 clock system 对外提供的接口：

```
1.  /*则加模块的引用计数*/
2.  struct clk *clk_get(struct device *dev, const char *id);
3.  /*减少模块的引用计数*/
4.  void clk_put(struct clk *clk);
5.  /*使能某个模块的时钟，比如 ADC 模块等*/
6.  int clk_enable(struct clk *clk);
7.  /*禁止模块的时钟*/
8.  void clk_disable(struct clk *clk);
9.  /*获得某类时钟频率*/
10.unsigned long clk_get_rate(struct clk *clk);
11./*设置某类部件的时钟(比如设置 CAM 接口时钟)*/
12.int clk_set_rate(struct clk *clk, unsigned long rate);
13./*获得父 clk*/
14.struct clk *clk_get_parent(struct clk *clk);
15./*设置父 clk*/
16.int clk_set_parent(struct clk *clk, struct clk *parent);
```

分享到：