

The logo features a green, textured, downward-pointing triangle. Inside the triangle, the words "FAR SIGHT" are written in white, with a red vertical line separating "FAR" and "SIGHT".

FAR SIGHT

嵌入式培训专家

S3C6410 ARM11 开发板

Linux BSP 构建

主讲：宋宝华

www.farsight.com.cn

今天的内容

✓BSP的组成部分

✓plat/mach各组件的实现

Ø 内核节拍

Ø 中断管理

Ø 时钟

Ø GPIO

Ø DMA

Ø IO内存映射

✓设备与资源

Ø platform device、resource和platform data

Ø uart/spi/i2c等设备板级resource

BSP的组成部分

✓BSP作用

- Ø 为内核的运行提供底层支撑
- Ø 屏蔽与板相关的硬件细节

✓基本组成

- Ø 时钟tick (HZ) 的产生
- Ø 系统中断控制的方法
- Ø GPIO、DMA、时钟资源的统一管理
- Ø 静态映射的IO内存
- Ø 设备的IO、中断、DMA等资源封装平台数据

ARM BSP的目录

vplat-xxx

linux-2.6/arch/arm/

plat-omap/

plat-pxa/

plat-s3c/

plat-s3c24xx/

plat-s3c64xx/

plat-stmp3xxx/

vmach-xxx

linux-2.6/arch/arm/

mach-s3c2400/

mach-s3c2410/

mach-s3c2412/

mach-s3c2440/

mach-s3c2442/

mach-s3c2443/

mach-s3c24a0/

mach-s3c6400/

mach-s3c6410/

时钟节拍的产生

▼ **sys_timer**和**timer_tick**

```
static irqreturn_t s3c2410_timer_interrupt(int irq, void *dev_id)
{
    timer_tick();
    return IRQ_HANDLED;
}

static struct irqaction s3c2410_timer_irq = {
    .name           = "S3C2410 Timer Tick",
    .flags           = IRQF_DISABLED | IRQF_TIMER | IRQF_IRQPOLL,
    .handler         = s3c2410_timer_interrupt,
};

static void __init s3c2410_timer_init(void)
{
    s3c2410_timer_resources();
    s3c2410_timer_setup();
    setup_irq(IRQ_TIMER4, &s3c2410_timer_irq);
}

struct sys_timer s3c24xx_timer = {
    .init            = s3c2410_timer_init,
    .offset           = s3c2410_gettimeoffset,
    .resume           = s3c2410_timer_setup
};
```

系统中断管理

▼ irq_chip

```
static struct irq_chip s3c_irq_uart = {
    .name          = "s3c-uart",
    .mask          = s3c_irq_uart_mask,
    .unmask        = s3c_irq_uart_unmask,
    .mask_ack      = s3c_irq_uart_maskack,
    .ack           = s3c_irq_uart_ack,
};

static void __init s3c64xx_uart_irq(struct uart_irq *uirq)
{
    for (offs = 0; offs < 3; offs++) {
        irq = uirq->base_irq + offs;
        set_irq_chip(irq, &s3c_irq_uart);
        set_irq_chip_data(irq, uirq);
        set_irq_handler(irq, handle_level_irq);
        set_irq_flags(irq, IRQF_VALID);
    }
    set_irq_chained_handler(uirq->parent_irq, s3c_irq_demux_uart);
}

void __init s3c64xx_init_irq(u32 vic0_valid, u32 vic1_valid)
{
    set_irq_chip(irq, &s3c_irq_timer);
    ...
    for (uart = 0; uart < ARRAY_SIZE(uart_irqs); uart++)
        s3c64xx_uart_irq(&uart_irqs[uart]);
}
```

✓ gpio_chip和统一的gpio_XXX API

```
struct gpio_chip {
    int      (*request)(struct gpio_chip *chip,
                        unsigned offset);
    void      (*free)(struct gpio_chip *chip,
                     unsigned offset);
    int      (*direction_input)(struct gpio_chip *chip,
                               unsigned offset);
    int      (*get)(struct gpio_chip *chip,
                   unsigned offset);
    int      (*direction_output)(struct gpio_chip *chip,
                                unsigned offset, int value);
    void      (*set)(struct gpio_chip *chip,
                    unsigned offset, int value);
};

int gpio_request(unsigned gpio, const char *label);
void gpio_free(unsigned gpio);
int gpio_direction_input(unsigned gpio);
int gpio_direction_output(unsigned gpio, int value);
int gpio_get_value_cansleep(unsigned gpio);
```

CLOCK管理

✓ 提供统一的clk_get、clk_put等API:

- Ø EXPORT_SYMBOL(clk_get);
- Ø EXPORT_SYMBOL(clk_put);
- Ø EXPORT_SYMBOL(clk_enable);
- Ø EXPORT_SYMBOL(clk_disable);
- Ø EXPORT_SYMBOL(clk_get_rate);
- Ø EXPORT_SYMBOL(clk_round_rate);
- Ø EXPORT_SYMBOL(clk_set_rate);
- Ø EXPORT_SYMBOL(clk_get_parent);
- Ø EXPORT_SYMBOL(clk_set_parent);

DMA管理

✓ 统一的DMA API支持：

- Ø `int request_dma(unsigned int chan, const char * device_id);`
- Ø `void free_dma(unsigned int chan);`
- Ø `void enable_dma(unsigned int chan);`
- Ø `void disable_dma(unsigned int chan);`
- Ø `void set_dma_mode (unsigned int chan, unsigned int mode);`
- Ø `void set_dma_sg (unsigned int chan, struct scatterlist *sg, int nr_sg);`

IO内存静态映射

▼ map_desc和iortable_init

```
static struct map_desc s3c_iodesc[] __initdata = {
    {
        .virtual    = (unsigned long)S3C_VA_SYS,
        .pfn        = __phys_to_pfn(S3C64XX_PA_SYSCON),
        .length      = SZ_4K,
        .type        = MT_DEVICE,
    }, {
        ...
    },
};

void __init s3c64xx_init_io(struct map_desc *mach_desc, int size)
{
    ...
    iortable_init(s3c_iodesc, ARRAY_SIZE(s3c_iodesc));
    iortable_init(mach_desc, size);
}
```

platform device和资源

✓ platform_device和资源

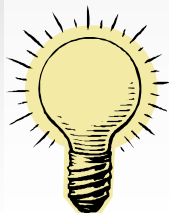
```
static struct resource smdk6410_smc911x_resources[] = {
    [0] = {
        .start = 0x18000000,
        .end   = 0x18000000 + SZ_64K - 1,
        .flags = IORESOURCE_MEM,
    },
    [1] = {
        .start = S3C_EINT(10),
        .end   = S3C_EINT(10),
        .flags = IORESOURCE_IRQ | IRQ_TYPE_LEVEL_LOW,
    },
};

static struct platform_device smdk6410_smc911x = {
    .name      = "smc911x",
    .id        = -1,
    .num_resources = ARRAY_SIZE(smdk6410_smc911x_resources),
    .resource   = &smdk6410_smc911x_resources[0],
};
```

✓ 提供与板相关的硬件设置数据

```
static struct smsc911x_platform_config smdk6410_smc911x_pdata = {
    .irq_polarity = SMSC911X_IRQ_POLARITY_ACTIVE_LOW,
    .irq_type     = SMSC911X_IRQ_TYPE_OPEN_DRAIN,
    .flags        = SMSC911X_USE_32BIT |
    SMSC911X_FORCE_INTERNAL_PHY,
    .phy_interface = PHY_INTERFACE_MODE_MII,
};

static struct platform_device smdk6410_smc911x = {
    ...
    .dev = {
        .platform_data = &smdk6410_smc911x_pdata,
    },
};
```



struct smsc911x_platform_config由对应设备的驱动定义，而 platform_data则由驱动引用。

SPI板级信息

vspi_board_info

```
static struct spi_board_info __initdata jive_spi_devs[] = {
    [0] = {
        .modalias      = "VGG2432A4",
        .bus_num       = 1,
        .chip_select    = 0,
        .mode           = SPI_MODE_3, /* CPOL=1, CPHA=1 */
        .max_speed_hz   = 100000,
        .platform_data  = &jive_lcm_config,
    }, {
        .modalias      = "WM8750",
        .bus_num       = 2,
        .chip_select    = 0,
        .mode           = SPI_MODE_0, /* CPOL=0, CPHA=0 */
        .max_speed_hz   = 100000,
    },
};
```

```
spi_register_board_info(jive_spi_devs, ARRAY_SIZE(jive_spi_devs));
```

I²C板级信息

vi2c_board_info

```
static struct i2c_board_info i2c_devs0[] __initdata = {
    { I2C_BOARD_INFO("24c08", 0x50), },
    { I2C_BOARD_INFO("wm8580", 0x1b), },

#ifdef CONFIG_SMDK6410_WM1190_EV1
    { I2C_BOARD_INFO("wm8350", 0x1a),
      .platform_data = &smdk6410_wm8350_pdata,
      .irq = S3C_EINT(12),
    },
#endif
};

static void __init smdk6410_machine_init(void)
{
    ...
    i2c_register_board_info(0, i2c_devs0, ARRAY_SIZE(i2c_devs0));
    i2c_register_board_info(1, i2c_devs1, ARRAY_SIZE(i2c_devs1));
    ...
}
```

MACHINE START

MACHINE_START(SMDK6410, "SMDK6410")

```
/* Maintainer: Ben Dooks <ben@fluff.org> */  
.phys_io    = S3C_PA_UART & 0xfff00000,  
.io_pg_offst= (((u32)S3C_VA_UART) >> 18) & 0xfffc,  
.boot_params    = S3C64XX_PA_SDRAM + 0x100,  
  
.init_irq     = s3c6410_init_irq,  
.map_io       = smdk6410_map_io,  
.init_machine  = smdk6410_machine_init,  
.timer        = &s3c24xx_timer,
```

MACHINE_END

范例：添加LDD6410板

U 修改Kconfig和Makefile:

Ø *linux-2.6.31/arch/arm/mach-s3c6410/Kconfig*

```
+      config MACH_LDD6410
+      bool "LDD6410"
+      select CPU_S3C6410
+      select S3C_DEV_FB
+      select S3C64XX_SETUP_FB_24BPP
+      help
+      Machine support for the LDD6410
+
+      config MACH_SMDK6410
+      bool "SMDK6410"
+      select CPU_S3C6410
```

Ø *linux-2.6.31/arch/arm/mach-s3c6410/Makefile*

```
+      obj-$(CONFIG_MACH_LDD6410) += mach-ldd6410.o
+      obj-$(CONFIG_MACH_SMDK6410) += mach-smdk6410.o
+      obj-$(CONFIG_MACH_NCP) += mach-ncp.o
```

U 增加新板子的文件:

linux-2.6.31/arch/arm/mach-s3c6410/mach-ldd6410.c

文档与参考实例

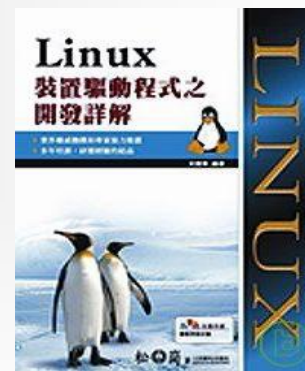
- ✓ Linux-2.6/arch/arm/
- ✓ <http://code.google.com/p/ldd6410/>
- ✓ 获取LDD6410源代码:

svn checkout *http*://ldd6410.googlecode.com/svn/trunk/ ldd6410-read-only

Linux设备驱动开发详解

✓ 主要出发点：

- ❑ 力求用最简单的实例讲解复杂的知识点，以免实例太复杂搅浑读者（驱动理论部分）
- ❑ 对Linux设备驱动多种复杂设备的框架结构进行了全面的介绍（驱动框架部分）
- ❑ 更面向实际的嵌入式工程，讲解开发必备的软硬件基础，及开发手段（调试与移植部分）
- ❑ 提供讨论与交流平台（华清远见，www.linuxdriver.cn）



✓ 主要出发点

- ✗ 开发LDD6410 SAMSUNG S3C6410开发板，所有实例均可在该板上直接运行和学习
- ✗ 全面升级为Linux 2.6.31内核，对Linux内核最新API和驱动子系统架构的变化进行介绍
- ✗ 对第一版中部分知识点进行整理和重新讲解
- ✗ 删除过时内容
- ✗ 新增大量内容：
 - n multi-touch触摸屏驱动
 - n CAMERA V4L2 驱动
 - n SPI主机和设备驱动
 - n ALSA SoC架构驱动
 - n USB 设备控制器 / gadget驱动 / USB OTG驱动
 - n SD / MMC驱动
 - n 内核移植 (BSP构建与开发)
 - n Android移植

华清远见Linux驱动课程

- ✓ 嵌入式Linux驱动初级班
- ✓ 通过本课程的学习，学员可以掌握Linux下字符设备、块设备、网络设备的驱动程序开发，同时掌握嵌入式Linux的系统开发和分析方法。
- ✓ 嵌入式Linux驱动开发高级班
- ✓ 本课程以案例教学为主，系统地介绍Linux下有关FrameBuffer、MMC卡、USB设备的驱动程序开发。
- ✓ 班级规模及环境
- ✓ 为了保证培训效果，增加互动环节，我们坚持小班授课，每期报名人数限15人，多余人员安排到下一期进行。人手一套开发板和开发用的PC主机。

华清远见

让我们一起讨论！



FAR_SIGHT

The logo features the word "FARSIGHT" in a white, serif font. A red, stylized, curved line separates the "FAR" and "SIGHT" parts of the word. The text is centered within a dark green, textured, downward-pointing triangle that has a 3D effect with lighter green highlights on its edges.

FARSIGHT

The success's road

www.farsight.com.cn

谢谢！