iTOP-4412-驱动-电源管理芯片 S5M8767 修改输出例程

本文档介绍，如何修改和控制 S5M8767，以 camera 扩展端子的 VDD28_AF，VDD28_CAM 为例，来具体介绍驱动中如何实现电源修改和控制。

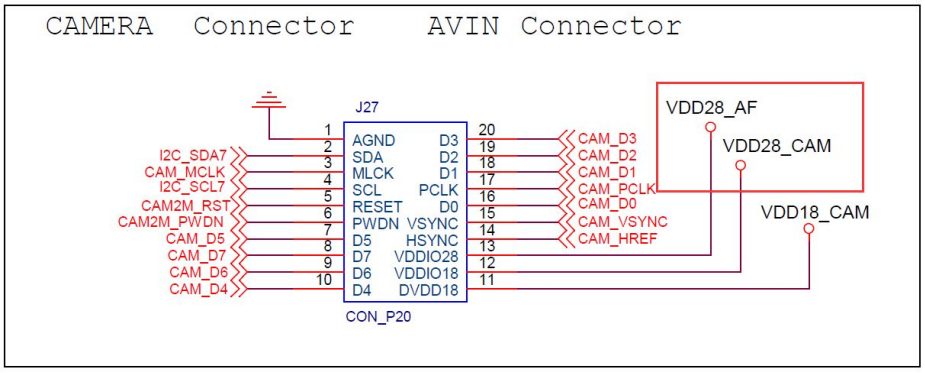
另外还有一个文档“iTOP-4412-驱动-电源管理芯片修改输出电压”，用户可以在技术支持群中搜到，其中涉及到具体结构的分析，也很有参考价值。

本文档以具体的驱动小例程介绍在已经配置好的源码中做修改，用户可以将其集成到自己的驱动中，也提供了驱动测试例程压缩包“power_s5m8767a.tar.gz”。

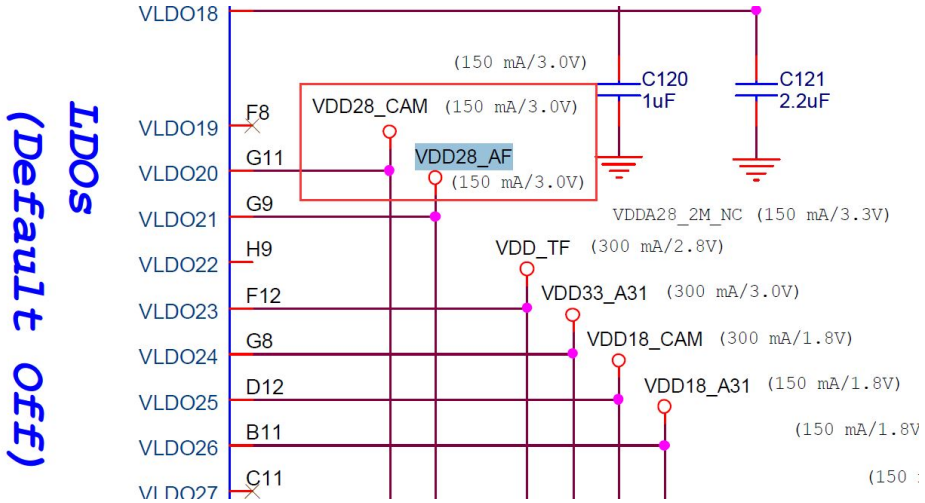
1 硬件分析

1.1 原理图分析

如下图所示，在底板原理图中找到 camera 扩展端子，这里以 VDD28_AF，VDD28_CAM 为例。camera 摄像头驱动中需要将其设置为 2.8v 的电压，后面我们将其修改为 3.3v 输出（需要去掉 camera 摄像头驱动）。



如下图所示，核心板原理图中搜索网络“VDD28_AF”和“VDD28_CAM”。可以看到“VDD28_AF”和“VDD28_CAM”分别对应电源芯片 S5M8767A 的“VLDO20”和“VLDO21”。



1.2 电源芯片 S5M8767A 的 datasheet 分析

S5M8767A 的 datasheet 的 2.3.1 小节，如下图所示。

2.3.1 LDO (P) 4, 5, 11, 12, 13, 14, 16, 17, 19, 20, 21, 22, 26, 27, 28 (150 mA, PMOS)

(VBAT = 3.7 V, T_A = 25 °C, unless otherwise specified)

Characteristics	Test Conditions		Min.	Typ.	Max.	Unit
Input voltage range (VINL) (1)			1.7	–	5.5	V
Under voltage Lockout	Rising, 100 mV Hysteresis			1.6	1.7	V
Battery Voltage Range	Equal or Higher than VINL		2.7	–	5.5	V
Output Voltage Range	IL = 150 mA Programmable in 50 mV steps		0.8	–	3.95	V
Default output voltage (VLDO)	LDO (P) 4, 5, 11, 13, 14, 16, 26, 27, 28	150 mA @VINL = VLDO +0.3V	–	1.8	–	V
	LDO (P) 17			2.8		
	LDO (P) 12, 19, 20, 21			3.0		
	LDO (P) 22			3.3		
Maximum Load Current	Normal Mode		150			mA
	Low-Power Mode		5			
Output Current Limit	VOUT = 90 % of VLDO		180	225	270	mA
Minimum Output Bypass Capacitance			0.7	1		µF
Ground Current	Battery Supply Current, with No Load	Shutdown		< 0.1		µA
		Normal Regulation		8	10	
		Low-Power Mode		0.5	1	
	Input Supply Current, with No Load	Shutdown		0	1	
		Normal Regulation		12	20	
		Low-Power Mode		2	5	

如上图所示，注意红框中的内容。最上面的红框中，表示输出的电流是 150mA，最低输出电压是 0.8v，最大电压是 3.95v。下面红框中，介绍的是默认输出电压，可以看到 LDO20 和 LDO21，默认输出的是 3.0v。

2 软件

如果要改变输出电压，可以通过修改平台文件实现；在驱动中，可以通过函数调用，控制电源输出。

通过前面的分析可知，ldo21 和 ldo20 输出电流范围是 0.8v 到 3.95v。

2.1 平台文件修改输出电压

在内核的“arch/arm/mach-exynos/mach-itop4412.c”文件中，如下图所示进行修改。

```
                REGULATOR_CHANGE_STATUS, 1); ///?
#if defined(CONFIG_VIDEO_OV5640) || defined(CONFIG_VIDEO_TVP5150)
REGULATOR_INIT(ldo20, "VDD28_CAM", 1800000, 1800000, 0,
                REGULATOR_CHANGE_STATUS, 1);
#else
//REGULATOR_INIT(ldo20, "VDD28_CAM", 2800000, 2800000, 0,
//              REGULATOR_CHANGE_STATUS, 1);
REGULATOR_INIT(ldo20, "VDD28_CAM", 3950000, 3950000, 0,
                REGULATOR_CHANGE_STATUS, 1);
#endif

/* modify by cym 20141106 */
#ifdef CONFIG_VIDEO_TVP5150
REGULATOR_INIT(ldo21, "VDD28_AF", 1800000, 1800000, 0,
                REGULATOR_CHANGE_STATUS, 1);
#else
//REGULATOR_INIT(ldo21, "VDD28_AF", 2800000, 2800000, 0,
//              REGULATOR_CHANGE_STATUS, 1);
REGULATOR_INIT(ldo21, "VDD28_AF", 3950000, 3950000, 0,
                REGULATOR_CHANGE_STATUS, 1);
#endif
REGULATOR_INIT(ldo22, "VDDA28_2M", 2800000, 2800000, 0,
                REGULATOR_CHANGE_STATUS, 1);
```

将

```
REGULATOR_INIT(ldo20, "VDD28_CAM", 2800000, 2800000, 0,
                REGULATOR_CHANGE_STATUS, 1);
```

注释掉，修改为 2800000,为 3950000（函数 REGULATOR_INIT 中的第一个参数表示 8767 电源芯片的第 20 路，第三个参数表示输出最低电压，第四个参数表示输出最高电压）。这里设置为最低和最高全部为 3.95v。同理，我们将第 21 路也修改为 3950000，如上图所示。

接着在 menuconfig 中，将 ov5640 摄像头的驱动去掉，因为在摄像头中会初始化和配置。ov5640 摄像头摄像头的配置路径如下图所示。下面截图是已经去掉的截图，默认缺省文件是配置上的。

```
Symbol: VIDEO_OV5640 [=n]
Type  : tristate
Prompt: OmniVision OV5640 sensor support
  Defined at drivers/media/video/Kconfig:1016
  Depends on: MEDIA_SUPPORT [=y] && VIDEO_CAPTURE_DRIVERS [=y] && I2C [=y] && VIDEO_V4L2 [=y]
  Location:
    -> Device Drivers
      -> Multimedia support (MEDIA_SUPPORT [=y])
        -> Video capture adapters (VIDEO_CAPTURE_DRIVERS [=y])
```

2.2 驱动例程

驱动例程“power_s5m8767a.tar.gz”和文档在同一压缩包中。

编写一个简单的驱动测试程序，源码如下所示。

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/i2c.h>
#include <linux/platform_device.h>
#include <linux/delay.h>
#include <linux/regulator/consumer.h>
#include <mach/gpio.h>
#include <plat/gpio-cfg.h>
#include <mach/regs-gpio.h>
#include <mach/regs-clock.h>
#include <linux/fs.h>
#include <linux/err.h>

struct regulator *ov_vddaf_cam_regulator = NULL;
struct regulator *ov_vdd5m_cam_regulator = NULL;
struct regulator *ov_vdd18_cam_regulator = NULL;
struct regulator *ov_vdd28_cam_regulator = NULL;


MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("iTOPEET_dz");

static int power(int flag)
{
```

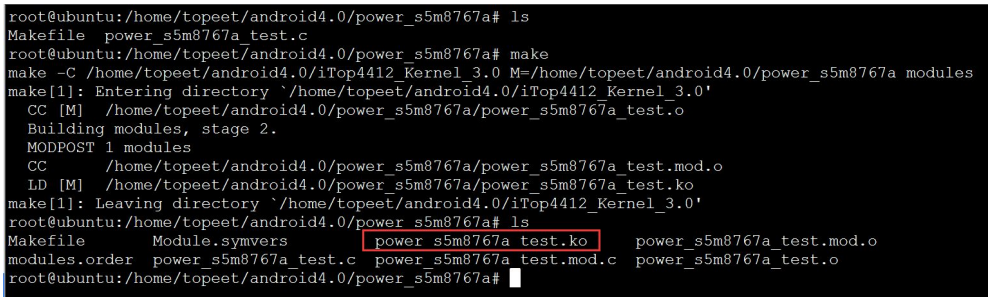
```
    if(1 == flag){
        regulator_enable(ov_vdd18_cam_regulator);
        udelay(10);
        regulator_enable(ov_vdd28_cam_regulator);
        udelay(10);
        regulator_enable(ov_vdd5m_cam_regulator); //DOVDD    DVDD 1.8v
        udelay(10);
        regulator_enable(ov_vddaf_cam_regulator);          //AVDD 2.8v
        udelay(10);
    }
    else if(0 == flag){
        regulator_disable(ov_vdd18_cam_regulator);
        udelay(10);
        regulator_disable(ov_vdd28_cam_regulator);
        udelay(10);
        regulator_disable(ov_vdd5m_cam_regulator);
        udelay(10);
        regulator_disable(ov_vddaf_cam_regulator);
        udelay(10);
    }

    return 0 ;
}

static void power_init(void)
{
    int ret;

    ov_vdd18_cam_regulator = regulator_get(NULL, "vdd18_cam");
    if (IS_ERR(ov_vdd18_cam_regulator)) {
        printk("%s: failed to get %s\n", __func__, "vdd18_cam");
        ret = -ENODEV;
        goto err_regulator;
    }

    ov_vdd28_cam_regulator = regulator_get(NULL, "vdda28_2m");
    if (IS_ERR(ov_vdd28_cam_regulator)) {
        printk("%s: failed to get %s\n", __func__, "vdda28_2m");
        ret = -ENODEV;
        goto err_regulator;
```

```
    }

    ov_vddaf_cam_regulator = regulator_get(NULL, "vdd28_af");
    if (IS_ERR(ov_vddaf_cam_regulator)) {
        printk("%s: failed to get %s\n", __func__, "vdd28_af");
        ret = -ENODEV;
        goto err_regulator;
    }

    ov_vdd5m_cam_regulator = regulator_get(NULL, "vdd28_cam");
    if (IS_ERR(ov_vdd5m_cam_regulator)) {
        printk("%s: failed to get %s\n", __func__, "vdd28_cam");
        ret = -ENODEV;
        goto err_regulator;
    }

err_regulator:
    regulator_put(ov_vddaf_cam_regulator);
    regulator_put(ov_vdd5m_cam_regulator);
    regulator_put(ov_vdd18_cam_regulator);
    regulator_put(ov_vdd28_cam_regulator);
}
static int hello_init(void)
{
    power_init();

    power(1);
    printk(KERN_EMERG "Hello World enter!\n");
    return 0;
}

static void hello_exit(void)
{
    power(0);

    printk(KERN_EMERG "Hello world exit!\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

Makefile 如下所示。

```
#!/bin/bash
obj-m += power_s5m8767a_test.o

KDIR := /home/topeet/android4.0/iTop4412_Kernel_3.0

PWD ?= $(shell pwd)

all:
	make -C $(KDIR) M=$(PWD) modules

clean:
	rm -rf *.o modules.order *.ko *mod.c Module.symvers
```

使用 make 命令编译驱动模块，如下图所示。



3 测试

如下图所示，加载驱动之后，测量电压大约为 2.85 左右（有压降），卸载驱动之后，电压为 0。说明驱动运行成功，用户在自己的项目中，假如需要用到电源控制，可以参考本例程来实现。

```
[root@iTOP-4412]# ls
power_s5m8767a_test.ko
[root@iTOP-4412]# insmod power_s5m8767a_test.ko

[   18.267564] Hello World enter!
[root@iTOP-4412]# rmmod power_s5m8767a_test
[   23.426302] Hello world exit!
```

联系方式

北京迅为电子有限公司致力于嵌入式软硬件设计，是高端开发平台以及移动设备方案提供商；基于多年的技术积累，在工控、仪表、教育、医疗、车载等领域通过 OEM/ODM 方式为客户创造价值。

iTOP-4412 开发板是迅为电子基于三星最新四核处理器 Exynos4412 研制的一款实验开发平台，可以通过该产品评估 Exynos 4412 处理器相关性能，并以此为基础开发出用户需要的特定产品。

本手册主要介绍 iTOP-4412 开发板的使用方法，旨在帮助用户快速掌握该产品的应用特点，通过对开发板进行后续软硬件开发，衍生出符合特定需求的应用系统。

如需平板电脑案支持，请访问迅为平板方案网“http://www.topeet.com”，我司将有能力为您提供全方位的技术服务，保证您产品设计无忧！

本手册将持续更新，并通过多种方式发布给新老用户，希望迅为电子的努力能给您的学习和开发带来帮助。

迅为电子

2017 年 12 月