linux内核中的GPIO系统之（1）：软件框架

作者：linuxer 发布于：2014-7-21 14:40 分类：GPIO子系统

一、前言

作为一个工作多年的系统工程师，免不了做两件事情：培训新员工和给新员工分配任务。对于那些刚刚从学校出来的学生，一般在开始的时候总是分配一些非常简单的任务，例如GPIO driver、LED driver。往往CPU datasheet的关于GPIO或者IO ports的章节都是比较简单的，非常适合刚入行的工程师。虽然GPIO子系统相关的硬件比较简单，没有复杂的协议，不过，对于软件抽象而言，其分层次的软件思想是每个嵌入式软件工程师需要掌握的内容。

我更倾向使用GPIO系统这个名字来代替GPIO driver这个名字，GPIO driver仅仅包含了pin signal状态控制和读取的内容，而GPIO系统包括了pin multiplexing、pin configuration、GPIO control、GPIO interrupt control等内容。本文主要是以3.14内核作为例子，讲述linux kernel中GPIO系统的软件框架。

二、GPIO相关硬件有哪些差异

嵌入式工程师总是要处理各种各样的target board，每个target board上的GPIO总是存在不同，例如：

1、和CPU的连接方式不同

对于ARM的嵌入式硬件平台，SOC本身可以提供大量的IO port，SOC上的GPIO controller是通过SOC的总线（AMBA）连接到CPU的。对于嵌入式系统而言，除了SOC的IO port，一些外设芯片也可能会提供IO port，例如：

（1）有些key controller芯片、codec或者PMU的芯片会提供I/O port

（2）有些专用的IO expander芯片可以扩展16个或者32个GPIO

从硬件角度看，这些IO和SOC提供的那些IO完全不同，CPU和IO expander是通过I2C（也有可能是SPI等其他类型的bus）连接的，在这种情况下，访问这些SOC之外的GPIO需要I2C的操作，而控制SOC上的GPIO只需要写寄存器的操作。不要小看这个不同，写一个SOC memory map的寄存器非常快，但是通过I2C来操作IO就不是那么快了，甚至，如果总线繁忙有可能阻塞当前进程，这种情况下，内核同步机制必须有所区别（如果操作GPIO可能导致sleep，那么同步机制不能采用spinlock）。

2、访问方式不同

SOC片内的GPIO controller和SOC片外的IO expander的访问当然不一样，不过，即便都是SOC片内的GPIO controller，不同的ARM芯片，其访问方式也不完全相同，例如：有些SOC的GPIO controller会提供一个寄存器来控制输出电平。向寄存器写1就是set high，向寄存器写0就是set low。但是有些SOC的GPIO controller会提供两个寄存器来控制输出电平。向其中一个寄存器写一就是set high，向另外一个寄存器写一就是set low。

3、配置方式不同

即便是使用了同样的硬件（例如都使用同样的某款SOC），不同硬件系统上GPIO的配置不同。在一个系统上配置为输入，在另外的系统上可能配置为输出。

4、GPIO特性不同。这些特性包括：

（1）是否能触发中断。对一个SOC而言，并非所有的IO port都支持中断功能，可能某些处理器只有一两组GPIO有中断功能。

（2）如果能够触发中断，那么该GPIO是否能够将CPU从sleep状态唤醒

（3）有些有软件可控的上拉或者下拉电阻的特性，有的GPIO不支持这种特性。在设定为输入的时候，有的GPIO可以设定debouce的算法，有的则不可以。

5、多功能复用

有的GPIO就是单纯的作为一个GPIO出现，有些GPIO有其他的复用的功能。例如IO expander上的GPIO只能是GPIO，但是SOC上的某个GPIO除了做普通的IO pin脚，还可以是SPI上clock信号线。

站内搜索

请输入关键词 搜索

功能

留言板
评论列表
支持者列表

最新评论

callme_friend
@pixiandouban：visio
callme_friend
@hit201j：太忙，没时间回复，需要的朋友可统一去以下链...
callme_friend
@icecoder：太忙，没时间回复，需要的朋友可统一去以下...
callme_friend
@就爱吃泡芙：太忙，没时间回复，需要的朋友可统一去以下链接下...
callme_friend
@今雨轩：太忙，没时间回复，需要的朋友可统一去以下链接下载：...
callme_friend
@xtydtc：太忙，没时间回复，需要的朋友可统一去以下链接...

文章分类

Linux内核分析(11)
统一设备模型(15)
电源管理子系统(42)
中断子系统(15)
进程管理(19)
内核同步机制(18)
GPIO子系统(5)
时间子系统(14)
通信类协议(7)
内存管理(27)
图形子系统(1)
文件系统(4)
TTY子系统(6)
u-boot分析(3)
Linux应用技巧(13)
软件开发(6)
基础技术(13)
蓝牙(16)
ARMv8A Arch(13)
显示(3)
USB(1)
基础学科(10)
技术漫谈(12)
项目专区(0)
X Project(28)

随机文章

基本电路概念之（二）：什么是电容?

三、硬件功能分类

ARM based SOC的datasheet中总有一个章节叫做GPIO controller（或者I/O ports）的章节来描述如何配置、使用SOC的引脚。虽然GPIO controller的硬件描述中充满了大量的寄存器的描述，但是这些寄存器的功能大概分成下面三个类别：

1、有些硬件逻辑是和IO port本身的功能设定相关的，我们称这个HW block为pin controller。软件通过设定pin controller这个硬件单元的寄存器可以实现：

（1）引脚功能配置。例如该I/O pin是一个普通的GPIO还是一些特殊功能引脚（例如memeory bank上CS信号）。

（2）引脚特性配置。例如pull-up/down电阻的设定，drive-strength的设定等。

2、如果一组GPIO被配置成SPI，那么这些pin脚被连接到了SPI controller，如果配置成GPIO，那么控制这些引脚的就是GPIO controller。通过访问GPIO controller的寄存器，软件可以：

（1）配置GPIO的方向

（2）如果是输出，可以配置high level或者low level

（3）如果是输入，可以获取GPIO引脚上的电平状态

3、如果一组gpio有中断控制器的功能，虽然控制寄存器在datasheet中的I/O ports章节描述，但是实际上这些GPIO已经被组织成了一个interrupt controller的硬件block，它更像是一个GPIO type的中断控制器，通过访问GPIO type的中断控制器的寄存器，软件可以：

（1）中断的enable和disable（mask和unmask）

（2）触发方式

（3）中断状态清除

四、如何通过软件抽象来掩盖硬件差异

传统的GPIO driver是负责上面三大类的控制，而新的linux kernel中的GPIO subsystem则用三个软件模块来对应上面三类硬件功能：

（1）pin control subsystem。驱动pin controller硬件的软件子系统。

（2）GPIO subsystem。驱动GPIO controller硬件的软件子系统。

（3）GPIO interrupt chip driver。这个模块是作为一个interrupt subsystem中的一个底层硬件驱动模块存在的。本文主要描述前两个软件模块，具体GPIO interrupt chip driver以及interrupt subsystem请参考本站其他相关文档。

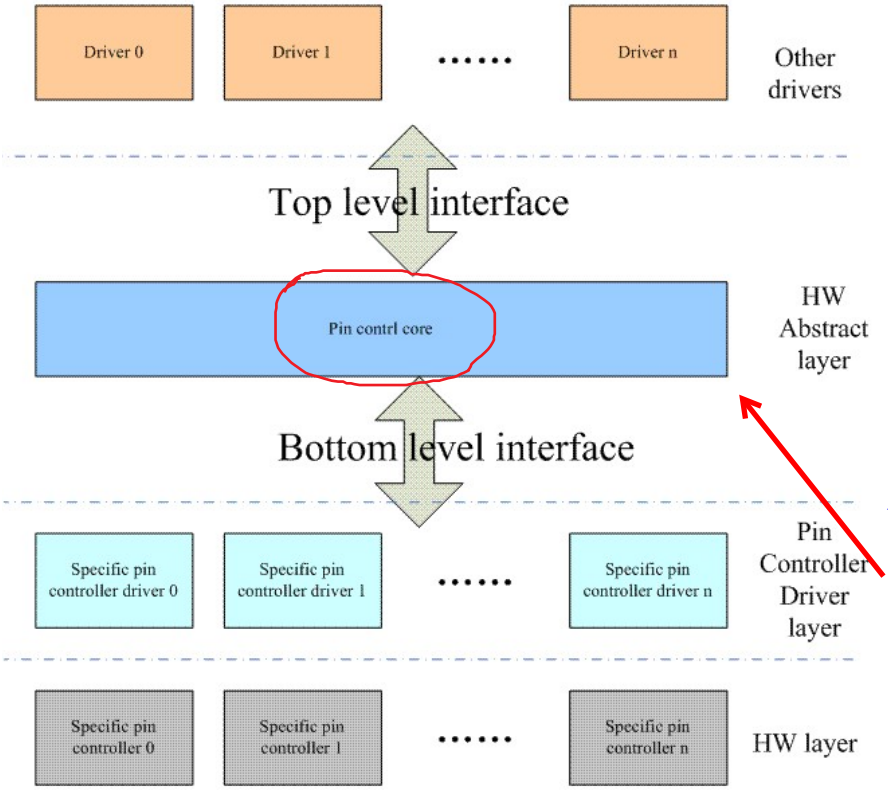
1、pin control subsystem block diagram

下图描述了pin control subsystem的模块图：

Linux cpufreq framework(2)_cpufreq driver
Linux I2C framework(2)_I2C provider
ARM64架构下地址翻译相关的宏定义
linux内核中的GPIO系统之（4）：pinctrl驱动的理解和总结

文章存档

2018年10月(1)
2018年8月(1)
2018年6月(1)
2018年5月(1)
2018年4月(7)
2018年2月(4)
2018年1月(5)
2017年12月(2)
2017年11月(2)
2017年10月(1)
2017年9月(5)
2017年8月(4)
2017年7月(4)
2017年6月(3)
2017年5月(3)
2017年4月(1)
2017年3月(8)
2017年2月(6)
2017年1月(5)
2016年12月(6)
2016年11月(11)
2016年10月(9)
2016年9月(6)
2016年8月(9)
2016年7月(5)
2016年6月(8)
2016年5月(8)
2016年4月(7)
2016年3月(5)
2016年2月(5)
2016年1月(6)
2015年12月(6)
2015年11月(9)
2015年10月(9)
2015年9月(4)
2015年8月(3)
2015年7月(7)
2015年6月(3)
2015年5月(6)
2015年4月(9)
2015年3月(9)
2015年2月(6)
2015年1月(6)
2014年12月(17)
2014年11月(8)
2014年10月(9)
2014年9月(7)
2014年8月(12)
2014年7月(6)
2014年6月(6)
2014年5月(9)
2014年4月(9)
2014年3月(7)
2014年2月(3)
2014年1月(4)



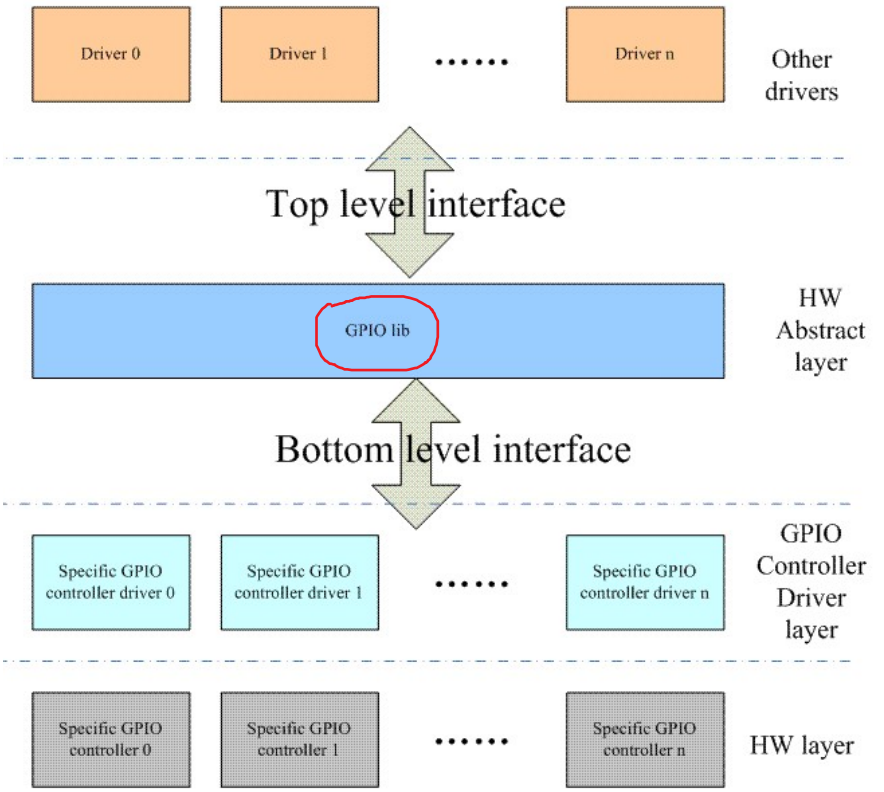
分层的软件设计思想

注册标准接口

底层的pin controller driver是硬件相关的模组，初始化的时候会向pin control core模块注册pin control设备（通过pinctrl_register这个bootom level interface）。pin control core模块是一个硬件无关模块，它抽象了所有pin controller的硬件特性，仅仅从用户（各个driver就是pin control subsystem的用户）角度给出了top level的接口函数，这样，各个driver不需要关注pin controller的底层硬件相关的内容。

2、GPIO subsystem block diagram

下图描述了GPIO subsystem的模块图：



基本上这个软件框架图和pin control subsystem是一样的，其软件抽象的思想也是一样的，当然其内部具体的实现不一样，我们会在后续的文章中描述。

*原创文章，转发请注明出处。蜗窝科技。*http://www.wowotech.net/io-port-control.html

标签: GPIO 软件框架

« Linux内核中的GPIO系统之（3）：pin controller driver代码分析 | ACCESS_ONCE宏定义的解释»

评论：

薛文旺
2018-06-21 09:40

你好！"如果操作GPIO可能导致sleep，那么同步机制不能采用spinlock"，为什么？

回复

smcdef
2018-06-21 22:41

@薛文旺：spinlock要求不能sleep

回复

floater
2017-12-01 17:33

请问有没有关于gpio扩展驱动的文章呢？扩展的gpio申请中断的原理看不明白，也不太清楚应该找那个子系统的文章看

回复

bigpillow
2017-11-13 11:05

Hi Linuxer,
我们这边有写一套完整的GPIO & pinctrl driver.,
但是一直没有找到很好的压测方案，
不知大神可否为测试整套GPIO function （大概200 pins）提供下思路
感谢~

回复

你想得美
2017-11-16 16:05

@bigpillow：你想得美

回复

wowo
2017-11-16 21:28

@你想得美：呵呵，还真没听过压测GPIO和pinctrl的，它们一般比较稳定，覆盖测一遍，对了就对了，不大可能还会出错。

回复

bigpillow
2017-11-24 11:09

@wowo：是，这边用户量比较大，怕万一被误踩register就尴尬了，所有要想办法压测。不过一直没有更好的方式。

回复

hongxiaoh
2017-01-14 19:56

你好，请问下，安卓linux 在kernel的键值表在哪里呢？

回复

hongxiaoh
2017-01-14 20:00

@hongxiaoh：linux,code = <> 里对应的键值表。

回复

wowo
2017-01-16 09:13

@hongxiaoh：不太明白您这里指的是什么。您可以把代码的前因后果贴出来。

回复

阿哈哈
2018-01-18 16:06

@hongxiaoh：generic.kl里面都有

回复

higari
2016-12-06 16:16

博主你好，我又来了。我有一个疑问特来请教（关于gpio的suspend/reusme）

我在my-gpio.c里写了两个函数
1.my_gpio_suspend
做的工作主要是把HW寄存器的值存入dir和data中。
2.my_gpio_resume
与suspend相反，把dir和data保存的值写回HW寄存器

执行步骤如下
echo 18 > /sys/class/gpio
echo out > /sys/class/gpio/gpio18/direction
echo 1 > /sys/class/gpio/gpio18/value
然后控制sys/power/state实现suspend/resume

于是问题是这样的:
resume后，上面做的这些设定（18,out,1）还会存在吗

因为手头上没有试验环境，上面又急着要设计方案。所以在此求助（没有实机摆弄的项目真是郁闷）

回复

wowo
2016-12-06 17:03

@higari：应该没有问题。

回复

higari
2016-12-06 18:01

@wowo：好的，谢谢啦。
另外，为什么你回复的那么快啊，一直盯着博客么

回复

wowo
2016-12-06 21:54

@higari：不客气，要注意suspend的时候ram不能掉电，否则变量里面保存的东西会丢。
PS：作为一个嵌入式工程师，怎么允许busy loop（一直盯着博客么）呢？怎么说也要中断触发（邮件提醒），否则定时器查询（偶尔看一下）啊！哈哈~~

回复

---

higari
2016-11-24 09:55

>总结的非常好啊，感谢你的分享
和博主的分享比起来真实微不足道，看了很多博主的文章受益匪浅。

博主要是有时间的话，能不能讲讲IOMMU,特别是核心IOMMU.C和ARM-SMMU.C这个驱动。最近要做ARM-SMMU.C的测试。没接触过这玩意，测试条目不知道怎么写了，正在啃那个300多页的HW MANUAL，英语苦手，看不太懂

回复

wowo
2016-11-24 10:18

@higari：以前有过一段时间我也想了解一下iommu，不过后来由于自己的外设的MMU做的不标准，就自己写了一个简单的，没有用kernel标准的框架。虽然想写，苦于没时间。不知道linuxer同学有时间写不。

回复

linuxer
2016-11-24 10:38

@wowo：写就写吧，反正最近一直酝酿一篇DMA mapping framework的文档，中间会涉及iommu的东西，但是不会太多的纠缠iommu的细节。

回复

wowo
2016-11-24 10:41

@linuxer：哈哈，不错，期待~ ~ ~

回复

---

higari
2016-11-21 11:18

博主你好，想请问一个问题
如果设置为输入口之后，进行gpio_set_value操作会发生什么？
gpiolib有没有对于这种user space的误操作情况采取什么手段。
网上查了查，没有什么头绪，所以想请教一下，嘿嘿

回复

wowo
2016-11-21 11:32

@higari：直接看一下代码，答案应该很快就出来了，然后顺便给我们分享一下:-)

回复

higari
2016-11-22 15:31

@wowo：>gpiolib有没有对于这种user space的误操作情况采取什么手段。
我直接看了gpiolib.c的代码，
_gpiod_set_raw_value

gpiod_set_raw_value
gpiod_set_value
这三个函数都没有check gpio的方向

>如果设置为输入口之后，进行gpio_set_value操作会发生什么？
我觉得对于设置为输入的gpio进行gpio_set_value操作，只是对它的输出端口的值进行设定，而不会影响gpio_get_value的结果，因为读的值是输入端口的值。
一个简单的例子就是gpio-rcar.c,他的gpio作为输入时和输出时，同样是gpio_get_value函数，但读取的寄存器是不一样的。
当gpio作为输入时，读取portin的值，输出时，读取portout.
另外不论gpio方向是输入还是输出，gpio_set_value都是对portout进行操作。
所以回到一开始的问题，方向设置为输入的gpio而言，gpio_set_value改变了是portout,而对于你真正想读取的portin并没有影响，当你再使用gpio_get_value时，总能读到你想要的正确的值。

以上有什么错误望指正，有别的疑问可以一起探讨，谢谢拉

回复

wowo
2016-11-22 16:07

@higari：总结的非常好啊，感谢你的分享！ ~ ~
所以kernel没有做任何处理，也是合理的了:-)

回复

---

emeralddream
2016-05-11 11:27

博主你好，gpio DTS 的配置中的宏定义有什么作用一直无法理解。
比如：
irq_gpio = <&gpio3 GPIO_A2 GPIO_ACTIVE_LOW>;这里 GPIO_ACTIVE_LOW 说明了低电平有效。

在驱动里首先会 gpio_request（）获取资源。然后 gpio_direction_input/output(),gpio_set_value()这2个 API去将 gpio 拉高 拉低。那么 DTS

回复

emeralddream
2016-05-11 11:28

@emeralddream：里的 GPIO_ACTIVE_LOW 究竟在解析的时候起到了说明作用？是不是对初始值有影响？

回复

emeralddream
2016-05-11 11:28

@emeralddream：里的 GPIO_ACTIVE_LOW 究竟在解析的时候起到了说明作用？是不是对初始值有影响？

回复

wowo
2016-05-11 13:11

@emeralddream：解析的时候没有什么用，就是把这个flag保存下来。
然后在后面使用的时候，比如xxx_power_enable，可以根据gpio的GPIO_ACTIVE_LOW flag，决定是把gpio拉低还是拉高。

回复

emeralddream
2016-05-11 13:22

@wowo：你所说的后面就是DTS里面?

回复

---

tom
2015-09-22 15:08

pin control subsystem 和 gpio subsystem 有什么区别，

回复

emeralddream
2016-03-31 09:04

@tom：个人理解 PINCTRL更高级吧，涉及到了pimux。目前gpio子系统只保留了API。具体的回调函数已经全部移到pinctrl子系统中去了。

回复

emeralddream
2016-03-31 09:04

@tom：个人理解 PINCTRL更高级吧，涉及到了pimux。目前gpio子系统只保留了API。具体的回调函数已经全部移到pinctrl子系统中去了。

回复

---

xie1230
2015-08-27 17:31

作者真的很牛，佩服，收藏了

回复

发表评论：

昵称

邮件地址 (选填)

个人主页 (选填)

发表评论