**SPRD GPIO User Guide**



**www.spreadtrum.com**

Spreadtrum Communications, Inc., Confidential and Proprietary

IMPORTANT NOTICE

COPYRIGHT NOTICE

**Copyright © 2013, Spreadtrum Communications, Inc. All rights reserved.**

TRADEMARKS

Spreadtrum Communications, Inc. and Spreadtrum Communications, Inc.’s products are exclusively owned by Spreadtrum Communications, Inc. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

WARRANTY DISCLAIMER

Spreadtrum Communications, Inc. makes no representations or warranties, either express or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

CONFIDENTIALITY

The information contained herein (including any attachments) is confidential. The recipient hereby acknowledges the confidentiality of this document, and except for the specific purpose, this document shall not be disclosed to any third party.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Data** | **Owner** | **Note** |
| 1.0 | 2014-6-7 | Andy zhang | First Version |
| 1.1 | 2014-6-23 | Andy zhang | Add two GPIO example |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

[Revision History 3](#_Toc391308083)

[1 General description 5](#_Toc391308084)

[2 SPRD ASIC spec 5](#_Toc391308085)

[2.1 IP version 5](#_Toc391308086)

[2.2 Spec path 5](#_Toc391308087)

[3 SPRD controller 6](#_Toc391308088)

[3.1 Pinmap 6](#_Toc391308089)

[3.1.1 上下拉配置 6](#_Toc391308090)

[3.1.2 Function功能选择 7](#_Toc391308091)

[3.1.3 驱动能力设置 7](#_Toc391308092)

[3.1.4 Sleep时的输入输出设置 7](#_Toc391308093)

[3.1.5 Sleep时的上下拉设置 7](#_Toc391308094)

[3.1.6 AP或CP的Sleep控制 8](#_Toc391308095)

[3.1.7 常见问题 8](#_Toc391308096)

[3.2 GPIO 11](#_Toc391308103)

[3.2.1 浮空&高阻 11](#_Toc391308106)

[3.2.2 Digital接口控制原则 12](#_Toc391308107)

[3.2.3 Pin配置原则 12](#_Toc391308108)

[3.2.4 GPIO中断 12](#_Toc391308109)

[4 SPRD driver 13](#_Toc391308110)

[4.1 Mocor 14](#_Toc391308111)

[4.1.1 GPIO 14](#_Toc391308112)

[4.1.2 IRQ 15](#_Toc391308113)

[4.2 u-boot 15](#_Toc391308114)

[4.3 Linux Kernel 16](#_Toc391308115)

[4.3.1 GPIO使用 16](#_Toc391308116)

[5 GPIO范例 18](#_Toc391308117)

[5.1 确定需求 18](#_Toc391308118)

[5.2 复用配置 19](#_Toc391308119)

[5.3 功能实现 20](#_Toc391308120)

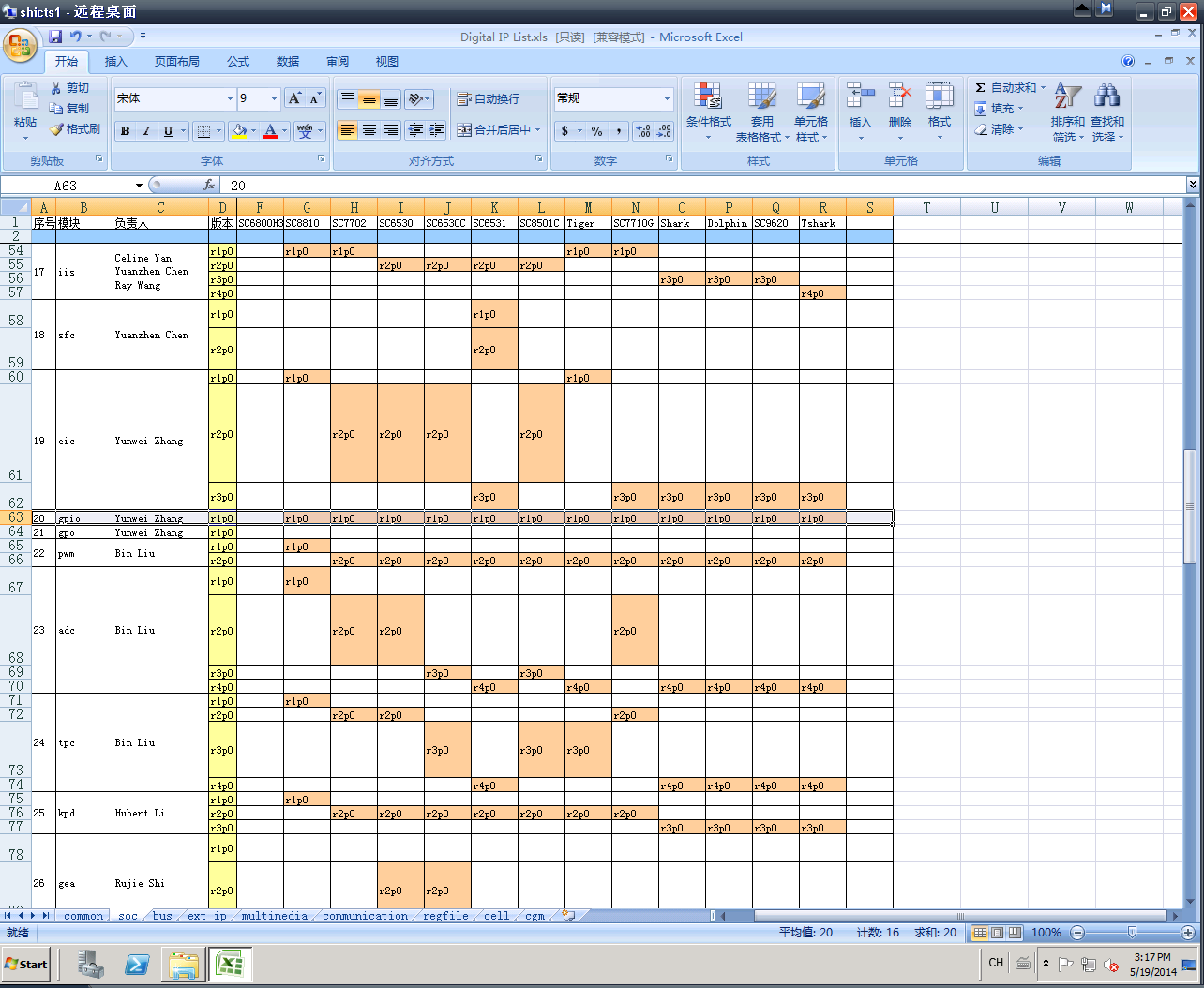
# General description

本文档描述SPRD PIN&GPIO，从硬件IP到驱动软件实现。

# SPRD ASIC spec

## IP version

GPIO IP version登录内网机器shicts1，到目录“D:\PROJECTS\DigitalIP\doc\Digital IP List.xls”文档，截至到项目tshark，gpio ip version均为r1p0，如下图示：



图表 1 ASIC doc

## Spec path

Pin spec path

[http://shsvn03/svn/PROJECTS/$(project\_name)/2.Engineering/2.7Interface/Device spec/Device Spec Chapter/ Pin Information Spec.doc](http://shsvn03/svn/PROJECTS/$(project_name)/2.Engineering/2.7Interface/Device%20spec/Device%20Spec%20Chapter/%20Pin%20Information%20Spec.doc) 没有这个路径的文件

gpio spec path

[http://shsvn03/svn/PROJECTS/$(project\_name)/2.Engineering/2.7Interface/Device spec/Device Spec Chapter/Micro-Controller Unit Peripheral Spec.doc](http://shsvn03/svn/PROJECTS/$(project_name)/2.Engineering/2.7Interface/Device%20spec/Device%20Spec%20Chapter/Micro-Controller%20Unit%20Peripheral%20Spec.doc)

注：$(project\_name)代表具体项目名称

# SPRD controller

GPIO与PIN关系紧密，经常需要一起来配置，这里一并列出。

## Pinmap

pinmap里面，包括：pin的

1. Function功能选择(Function 0~3)
2. 上下拉设置(WPD ,WPU , x )
3. 驱动能力设置(DS 0~3)
4. Sleep时的上下拉设置(WPD,WPU,X)
5. Sleep时的输入输出设置(Input,Output,Hiz)
6. 强上拉的设置(WPUS, x)
7. AP或CP的Sleep控制--AP+CP架构(AP,CP0~CP2)
8. 其他(PIN\_CTRL0~3)

### 上下拉配置

PU/PD代表系统在Active时（DeepSleep以外），IO的上拉和下拉的选择。有WPUS，WPU，WPD和x四种选择。

|  |  |
| --- | --- |
| 配置 | 说明 |
| WPUS | IO的强上拉电阻，一般在4.7K，会随着电压降低增大。一般适用于I2C/SIM\_IO/RF\_SDA/SDIO/EMMC等协议需要的地方。 |
| WPU | IO的上拉电阻，一般在70K，会随着电压降低增大。适用于输入引脚，无外部上下拉。默认需要保持高电平，低电平有效的GPIO或者中断功能。  或者外部连接信号会出现高阻态/输入态。  After Reset是ie=1,WPU的NC引脚。 |
| WPD | WPD—IO的下拉电阻，一般在70K，会随着电压降低增大。适用于输入引脚，无外部上下拉。默认需要保持低电平，高电平有效的GPIO或者中断功能。  或者外部连接信号会出现高阻态/输入态。  After Reset是ie=1,WPD的NC引脚。 |
| X | IO是输出，或者外部有上下拉。或者输入引脚IO外部连接的器件信号固定（不是高阻或者半高电平）。  Clock输入或者输出  After Reset是oe=1的NC引脚。 |

### Function功能选择

一般情况下Function功能是4种。Function是从0开始，3结束。一般Function 3是GPIO功能。Function是绝对不能选错的。Type代表这个Function的IO可以设置的类型。I—输入，O—输出，T—三态/高阻。了解这个类型有助于分析问题。



图表 2 PIN function

### 驱动能力设置

一般情况下Drive Strength功能是4种。2/6/12/24mA对应驱动能力0/1/2/3。3为最强。一般情况下驱动能力(DS)设置为1就可以。部分引脚比如CLK引脚需要大一些的选择2或者3。驱动能力越大，信号过冲越大。

### Sleep时的输入输出设置

SlpOE/IE代表系统在DeepSleep时，IO输入输出选择。有Input，Output和Hiz三种选择。

|  |  |
| --- | --- |
|  |  |
| Input | 适用于需要唤醒系统的IO中断引脚。或者AP进入Sleep还需要输入给CP或者音频的引脚，比如FM\_Dx/PCM/IIS等RTC32K输入。  其他需要保持function状态的引脚。 |
| Output | 适用于DeepSleep要保持输出状态的引脚，输出高低由进入Sleep之前的状态决定。  比如部分Reset引脚，电源的Enable信号，RFCTL，Power Down信号等。  RTC32K时钟输出。 |
| Hiz | NC引脚  Deepsleep不需要唤醒 |

### Sleep时的上下拉设置

SlpPU/PD代表系统在DeepSleep时，IO的上拉和下拉的选择。有WPU，WPD和x三种选择。

|  |  |
| --- | --- |
| 配置 | 说明 |
| SLEEP WPU | Sleep设置为Hiz或Input，外部需要高电平或者上拉的引脚。比如I2C/SIM\_IO/Key\_IN/Nand R/B等。 |
| SLEEP WPD | Sleep设置为Hiz或Input，外部需要低电平或者下拉的引脚。或者Hiz的NC引脚。 |
| SLEEP X | Sleep时为输出的引脚。 |

### AP或CP的Sleep控制

CP or AP代表IO随哪个AP或者CP进入Sleep而转变为Sleep状态。这个设置只在AP+CP的架构里面有，虚拟机和功能机没有这个设置。

|  |  |
| --- | --- |
|  |  |
| AP | 只有AP会使用的IO，或者NC的IO。不含音频相关的。 |
| CP0或者CP1 | 跟W/TD/GSM/LTE等有关的引脚，比如RF\_SPI/RFCTL和SIM。 |
| CP2 | 跟BT/WIFI（SR2351方案）有关的控制引脚，RF SPI等。不确定的设置成AP|CP2。 |
| AP|CPx  CPx|CPx  ALL  AP|CPx|CPx  DCP | 音频接口一般设置为ALL，或者AP|CPx的形式。Adie到Ddie的接口也设置为ALL。  如果多模手机，RFCTL/SIM卡共用，应该也设置成CP0|CP1的设置。  不确定的也采用ALL。 |

### 常见问题

表格 1Pin Sleep配置原则推荐表

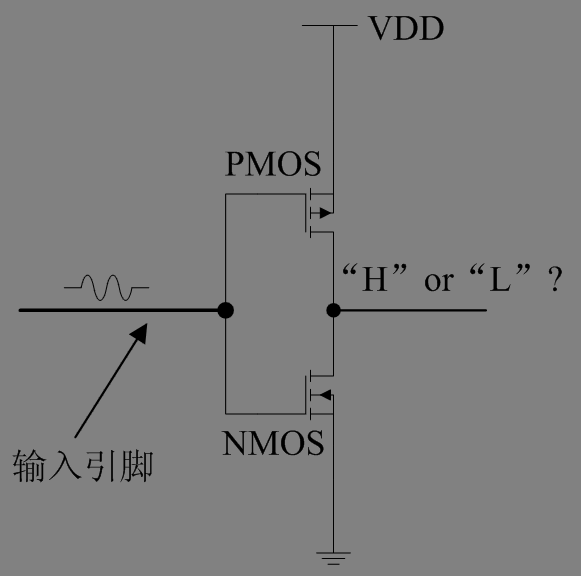
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SIE | SOE | SPU | SPD | Description |
| 0 | 0 | 0 | 0 | 外接器件或电路在Deepsleep时断电，最省电状态 |
| 0 | 0 | 0 | 1 | 外接器件或电路在Deepsleep时有电，外部无Pull-up，且低电平对外部器件的状态不会有影响，此时需要将Pin拉到一个稳定的状态 |
| 0 | 0 | 1 | 0 | 外接器件或电路在Deepsleep时有电，外部无Pull-down，且高电平对外部器件的状态不会有影响，此时需要将Pin拉到一个稳定的状态 |
| 0 | 0 | 1 | 1 | Forbidden |
| 0 | 1 | 0 | 0 | Deepsleep时需要输出一个固定电平的Pin，如GPIO配置为输出 |
| 0 | 1 | 0 | 1 | Not Recommend，如果内部在Deepsleep时输出高电平，将产生漏电回路 |
| 0 | 1 | 1 | 0 | Not Recommend，如果内部在Deepsleep时输出低电平，将产生漏电回路 |
| 0 | 1 | 1 | 1 | Forbidden |
| 1 | 0 | 0 | 0 | Pin的功能为可唤醒的信号或中断，外部器件或电路已有Pull-up或Pull-down |
| 1 | 0 | 0 | 1 | Pin的功能为可唤醒的信号或中断，高电平或上升沿有效，外部无Pull-up |
| 1 | 0 | 1 | 0 | Pin的功能为可唤醒的信号或中断，低电平或下降沿有效，外部无Pull-down |
| 1 | 0 | 1 | 1 | Forbidden |
| 1 | 1 | 0 | 0 | Not Recommend |
| 1 | 1 | 0 | 1 | Not Recommend |
| 1 | 1 | 1 | 0 | Not Recommend |
| 1 | 1 | 1 | 1 | Forbidden |

SIE/SOE：Sleep模式下的输入输出使能 SPU/SPD：Sleep模式下的弱上下拉设置

此表为pm\_func[ ]中相关参数的配置原则，绿色底色为推荐方式。



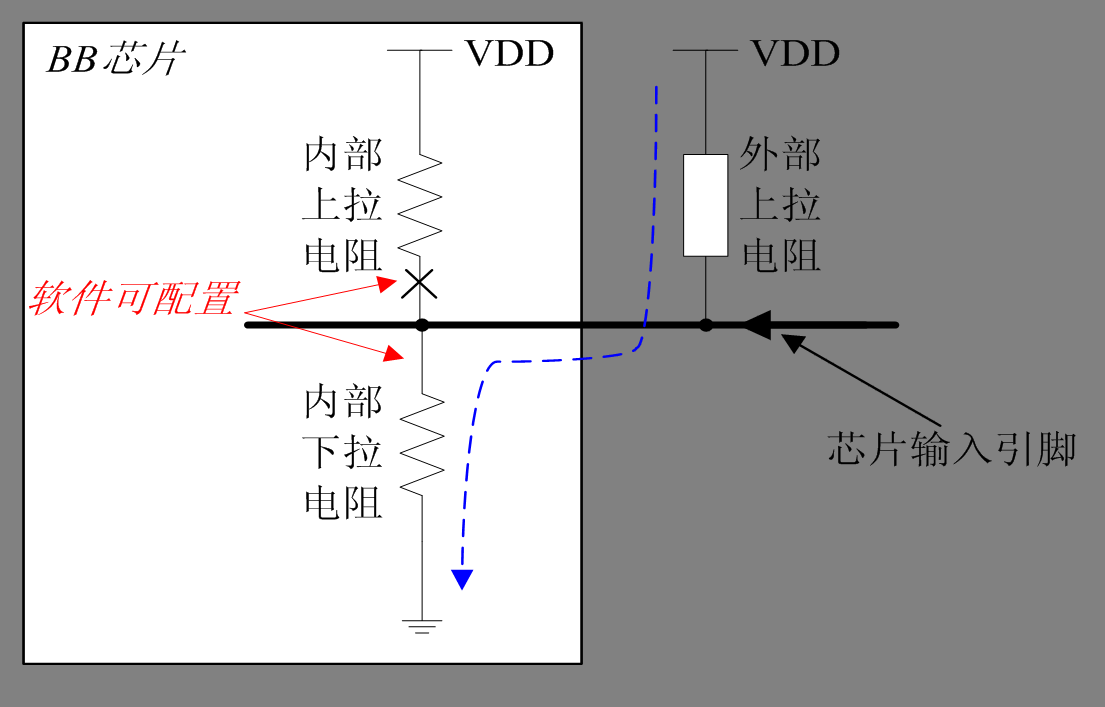
#### 悬空输入引起漏电



图表 3 浮空脚引起漏电原理

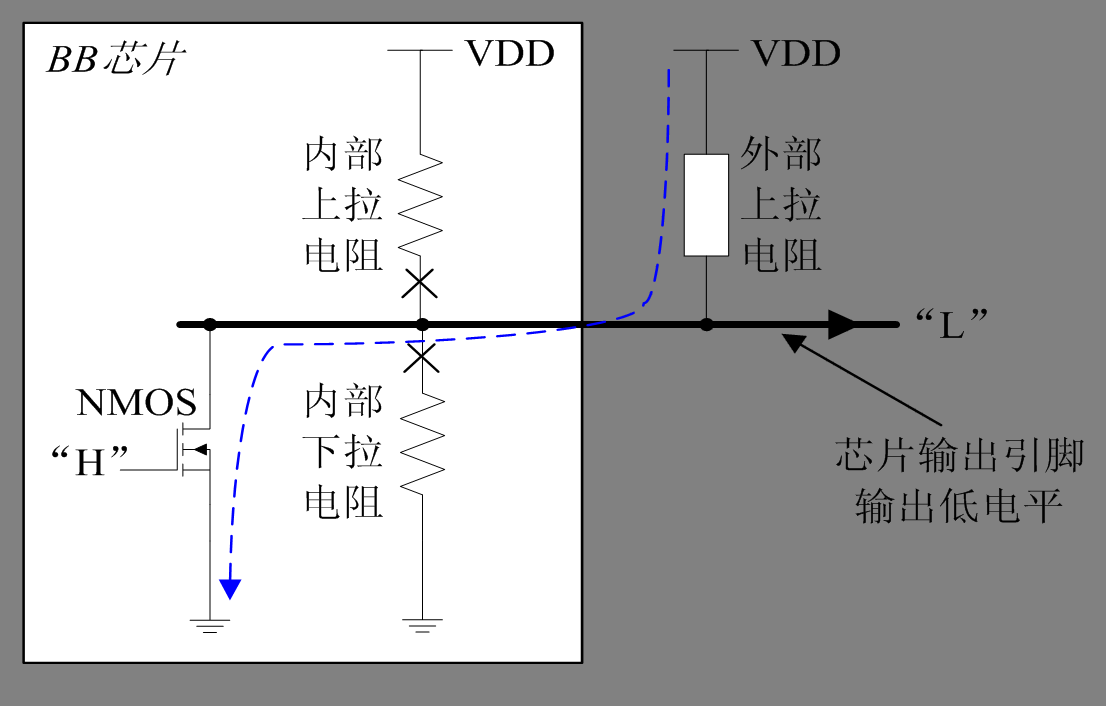
输入电平在中间电平时，耗电将达最大，因为此时上下两个MOS都处于半导通状态。

#### 输入引脚漏电



图表 4 输入引脚漏电案例

#### 输出引脚漏电

****

图表 5 输出引脚漏电案例

## GPIO

GPIO Pin配置的影响

1. 功耗：静态衬底电流
2. 维持外围电路正常工作
3. 系统工作稳定性
4. Gpio配置的影响
5. 中断配置的合理性，影响系统稳定性（一般需结合Pin配置）

### 浮空&高阻

* 悬空（浮空，floating）：就是逻辑器件的输入引脚即不接高电平，也不接低电平。一般实际运用时，引脚不建议悬空，易受干扰。（IE=1,OE=0, PD=0,PU=0）
* 高阻态这是一个数字电路里常见的术语，指的是电路的一种输出状态，既不是高电平也不是低电平，如果高阻态再输入下一级电路的话，对下级电路无任何影响，和没接一样，如果用万用表测的话有可能是高电平也有可能是低电平，随它后面接的东西定。 （IE=0,OE=0, PD=0,PU=0）

### Digital接口控制原则

* 芯片的Core电压一般工作在1.8V（1.2V），而Memory接口、IO接口一般工作在3.0V（1.8V），因此在Memory接口、IO接口电路和芯片内部Core电路之间存在电平转换电路，如果接口信号在中间电平停留的时间过长，会导致电平转换电路中上下两个MOS管均处于打开状态，从而导致电流漏电，最终在Deep Sleep电流波形上产生电流尖峰和鼓包。
* 从这个角度出发，Digital接口信号控制的一般原则：
  + Deep Sleep后，Digital接口信号禁止处于中间电平或三态；
  + Deep Sleep前后不应存在电平转换；如有确实需要的电平转换过程，则应设法加快电平转换的时间；

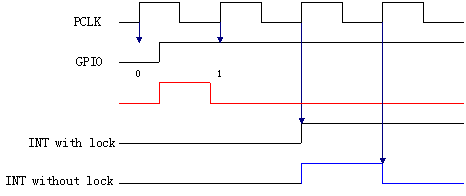
### Pin配置原则

* 下列原则是，基于PAD电压域打开并与外设匹配前提下
  + PU/PD设置，需要注意与外围不可形成回路，这在Wake/Sleep状态都需要考虑.即若引脚的内部或外部设置了下拉电阻，则清除上拉电阻配置，避免产生电流回路，反之亦然
  + 输入Pin不能浮空，如果外围无上下拉，需要使用芯片集成PU/PD
  + 需要在sleep保持输出Pin，OE=1
  + 需要在sleep保持输入Pin，IE=1
  + 无特殊情况，FPU/FPD与SPU/SPD保持一致
  + 可通过设置IE=0,OE=0实现硬件隔离
* 特殊情况
  + 外设供电被断电，而与BB相连的总线仍然被供电，会存在透过总线进入器件与VDD形成漏电回路。此时建议设置IE=0,OE=0,SPU=0,SPD=0作隔离
  + 使用Pin时需要确保所在电压域打开

### GPIO中断

GPIO中断的触发方式分为电平触发和边沿触发两种，判别原理各不相同，简要说明如下：

#### 边沿触发



图表 6 GPIO边沿触发时序图

当GPIO的沿中断开启之后，系统会在每个PCLK周期采样GPIO信号，并根据两个相邻时钟信号采样到的GPIO电平是否相同来决定是否产生了沿中断。如果相邻时钟信号采样到的电平不同，则认为有一个上升（或者下降）沿中断，并在下一个时钟周期产生中断信号。但是如果GPIO信号过窄，不足以维持到下个采样点的到来（如图中红线所示），便认为是毛刺信号，并且不会产生中断。这种问题是无法解决的。

在实际的芯片设计中，中断信号上通常都会增加锁存，从而使中断信号一直保持下去，直到通过寄存器操作清除相应的中断状态位为止。而没有锁存的中断信号则会在下一个时钟周期到来时恢复默认值（即低电平，如图中蓝线所示），这样就有可能发生系统还没有来得及响应这个中断信号，信号就已经消失的情况，并最终导致中断的丢失。我们的所有芯片实际上都有锁存，因此，不用担心产生中断丢失的情况。

注：在deepsleep状态，由于PCLK已经关闭，不可能去判别GPIO的沿中断，因此系统在这种状态下只允许电平触发方式，而禁止边沿触发方式。

#### 电平触发

这种触发方式下，GPIO信号被直接连接到中断信号的逻辑电路上，也就是说，当GPIO信号变为所设定的触发电平的同时，中断信号也会随即产生，并且一直保持到GPIO恢复原始电平为止。

电平触发中断没有锁存，如果触发电平维持时间比较短，系统就有可能检测不到这个中断信号，从而发生前面提到的中断丢失的情况，因此，在软件设计时，要特别注意所提供的GPIO控制信号会不会太窄，导致系统没有足够的时间去响应它；以及系统的中断处理程序耗时是不是太长以至来不及处理其他的中断信号等问题。

# SPRD driver

本章介绍GPIO驱动。

1. GPIO的配置一般分为两部分，第一部分是pinmap的配置信息，pinmap配置决定了pin的复用功能选择，上下拉，驱动能力，sleep配置。这部分功能一般在u-boot中配置。
2. 配置gpio控制器，来设置GPIO属性。这部分功能在kernel中配置。
   1. 普通输入输出方向，使能。
   2. 中断，首先需要配置为输入，触发中断使能，触发中断方式（电平触发，边沿触发）。

## Mocor

Mocor平台GPIO代码位于目录chip\_drv\chip\_module\gpio\下，分为hal层和phy层，hal层对应于外部接口的通用抽象，phy层则对应于ip版本。

### GPIO

GPIO接口

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // Description: This function used to set the value of a gpio pin  // Author: Lin.Liu  // Note:  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  PUBLIC void GPIO\_SetValue **(**uint32 gpio\_id**,** BOOLEAN b\_on**)**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // Description: This function used to get the value of a gpio pin  // Author: Lin.Liu  // Note:  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  PUBLIC BOOLEAN GPIO\_GetValue **(**uint32 gpio\_id**)**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // Description: This function used to get the direction of a gpio pin  // Note: 0 - Input  // 1 - Output  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  PUBLIC BOOLEAN GPIO\_GetDirection **(**uint32 gpio\_id**)**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // Description: This function used to set the direction of a gpio pin  // Note: 0 - Input  // 1 - Output  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  PUBLIC void GPIO\_SetDirection **(**uint32 gpio\_id**,** BOOLEAN directions**)** |

### IRQ

Mocor平台，GPIO驱动维护一个向量表，支持hisr。GPIO中断发生后，根据GPIO号进行二次中断分发，处理。

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // Description: This function Register gpio interrupt handler.  // Author: Steve.Zhan  // Note:  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  LOCAL uint32 GPIO\_RegInt **(**void**)**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // Description: This function be called by interrupt handler when is  // GPIO interrupt.  // Author: Xueliang.Wang  // Note: PUBLIC this function for gpio config file only.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  PUBLIC ISR\_EXE\_T GPIO\_Handler **(**uint32 int\_num**)**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // Description: This function is gpio hisr handler  // Author: Xueliang.Wang  // Note: PUBLIC this function for gpio config file only.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  LOCAL void GpioHisrFunc **(**uint32 cnt**,** void **\***pData**)** |

## u-boot

pin的配置都是在u-boot中完成，包括pin的上下拉，sleep上下拉，驱动能力，复用功能等。建议把不需要动态配置的pin放在这里统一配置，需要动态配置的，这里需要配置好初始状态，并注明kernel中动态配置。

|  |
| --- |
| static pinmap\_t pinmap**[]={**  **{**REG\_PIN\_CTRL0**,(**1**<<**20/\*gps real select\*/**)|(**1**<<**26/\*RFSDA2 source select function 2 of U0TXD\*/**)|**0**},**  **{**REG\_PIN\_CTRL1**,**0**},**  **{**REG\_PIN\_CTRL2**,(**1**<<**27/\*rfsbi cp0 control cp1 pin\*/**)|**0**},**  **{**REG\_PIN\_CTRL3**,(**1**<<**30/\*bt i2s internal connection enable\*/**)|**0**},**  **{**REG\_PIN\_CTRL4**,(**1**<<**6/\*vsd1\_ms\*/**)|**0**},**  **{**REG\_PIN\_CTRL5**,**0**},**  **{**REG\_PIN\_TRACECLK**,** BIT\_PIN\_SLP\_AP**|**BIT\_PIN\_NULL**|**BITS\_PIN\_DS**(**1**)|**BITS\_PIN\_AF**(**3**)|**BIT\_PIN\_WPD**|**BIT\_PIN\_SLP\_WPD**|**BIT\_PIN\_SLP\_Z**},**  **{**REG\_PIN\_TRACECTRL**,** BIT\_PIN\_SLP\_AP**|**BIT\_PIN\_NULL**|**BITS\_PIN\_DS**(**1**)|**BITS\_PIN\_AF**(**3**)|**BIT\_PIN\_WPD**|**BIT\_PIN\_SLP\_WPD**|**BIT\_PIN\_SLP\_Z**},**  **{**REG\_PIN\_TRACEDAT0**,** BIT\_PIN\_SLP\_AP**|**BIT\_PIN\_NULL**|**BITS\_PIN\_DS**(**1**)|**BITS\_PIN\_AF**(**3**)|**BIT\_PIN\_WPD**|**BIT\_PIN\_SLP\_WPD**|**BIT\_PIN\_SLP\_Z**},**  **{**REG\_PIN\_TRACEDAT1**,** BIT\_PIN\_SLP\_AP**|**BIT\_PIN\_NULL**|**BITS\_PIN\_DS**(**1**)|**BITS\_PIN\_AF**(**3**)|**BIT\_PIN\_WPD**|**BIT\_PIN\_SLP\_WPD**|**BIT\_PIN\_SLP\_Z**},**  **{**REG\_PIN\_TRACEDAT2**,** BIT\_PIN\_SLP\_AP**|**BIT\_PIN\_NULL**|**BITS\_PIN\_DS**(**1**)|**BITS\_PIN\_AF**(**3**)|**BIT\_PIN\_WPD**|**BIT\_PIN\_SLP\_WPD**|**BIT\_PIN\_SLP\_Z**},**  **{**REG\_PIN\_TRACEDAT3**,** BIT\_PIN\_SLP\_AP**|**BIT\_PIN\_NULL**|**BITS\_PIN\_DS**(**1**)|**BITS\_PIN\_AF**(**3**)|**BIT\_PIN\_WPD**|**BIT\_PIN\_SLP\_WPD**|**BIT\_PIN\_SLP\_Z**},**  **{**REG\_PIN\_TRACEDAT4**,** BIT\_PIN\_SLP\_AP**|**BIT\_PIN\_NULL**|**BITS\_PIN\_DS**(**1**)|**BITS\_PIN\_AF**(**3**)|**BIT\_PIN\_WPD**|**BIT\_PIN\_SLP\_WPD**|**BIT\_PIN\_SLP\_Z**},**  ...  **}** |

## Linux Kernel

在kernel中gpio驱动目录为”drivers/gpio”.框架层文件gpiolib.c，其中有对gpio一些结构的封装，我们的驱动需要实现相应的底层接口，注册到系统即可。

### GPIO使用

#### GPIO

常用GPIO接口：

|  |
| --- |
| /\*allocate gpio\*/  int gpio\_request**(**unsigned gpio**,** const char **\***label**);**  /\* set as input or output, returning 0 or negative errno \*/  int gpio\_direction\_input**(**unsigned gpio**);**  int gpio\_direction\_output**(**unsigned gpio**,** int value**);**  /\* GPIO INPUT: return zero or nonzero \*/  int gpio\_get\_value**(**unsigned gpio**);**  /\* GPIO OUTPUT \*/  void gpio\_set\_value**(**unsigned gpio**,** int value**);**  /\*free\*/  gpio\_free**(**unsigned gpio**);** |

伪代码

|  |
| --- |
| int xxx\_gpio\_request**(...)**  **{**  **...**  ret **=** gpio\_request**(**gpio**,** "label"**);**  **if** **(**ret**)** **{**  pr\_warn**(**"Cannot get gpio %d\n"**,** sel\_a**);**  **goto** fail**;**  **}**  gpio\_direction\_output**(**gpio**,** 1**);**  **...**  fail**:**  gpio\_free**(**gpio**);**  **...**  **return;**  **}** |

#### IRQ

GPIO映射为irq中断：

|  |
| --- |
| /\* map GPIO numbers to IRQ numbers \*/  int gpio\_to\_irq**(**unsigned gpio**);**  /\* map IRQ numbers to GPIO numbers (avoid using this) \*/  int irq\_to\_gpio**(**unsigned irq**);** |

伪代码

|  |
| --- |
| int xxx\_gpio\_irq**(...)**  **{**  **...**  ret **=** gpio\_request**(**gpio**,** "gpio\_irq"**);**  **if** **(**ret**)** **{**  pr\_warn**(**"Cannot get gpio %d\n"**,** sel\_a**);**  **goto** fail**;**  **}**  gpio\_direction\_input**(**gpio**);**  irq **=** gpio\_to\_irq**(**gpio**);**  error **=** request\_irq**(**irq**,** handler**,** IRQ\_FLAG**,** "irq\_name"**,** **NULL);**  **if** **(**error**)**  printk**(**KERN\_ERR "%s: could not request IRQ "**,** \_\_func\_\_**);**  fail**:**  gpio\_free**(**gpio**);**  **...**  **return;**  **}** |

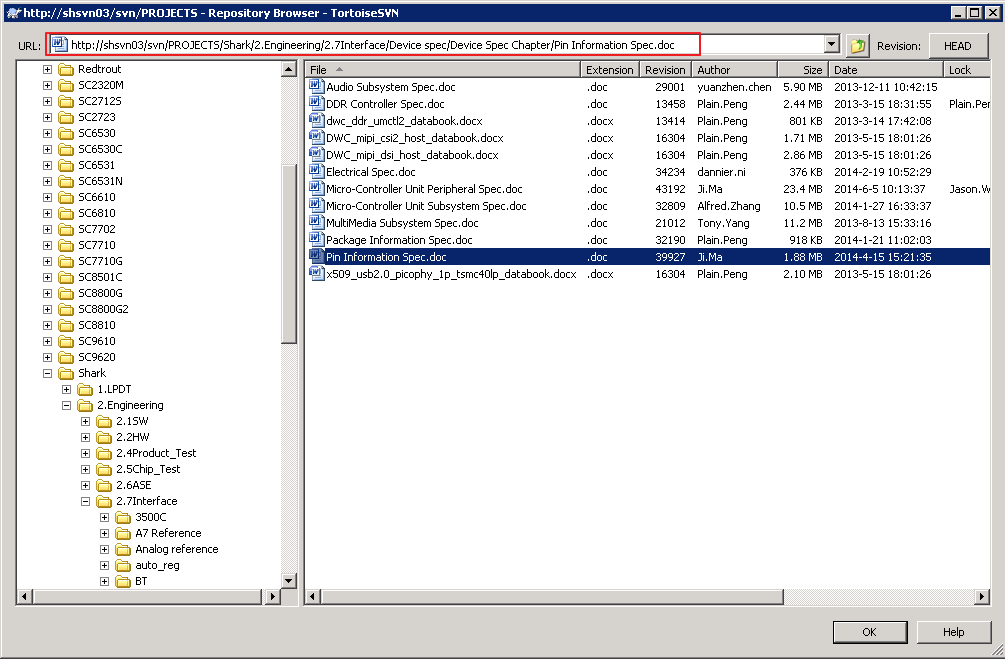
# GPIO范例

本节以sp8830 sprd\_sensor驱动以及touchscreen驱动为例，给出GPIO从启动到kernel驱动中的整个流程的使用范例。其中sprd\_sensor驱动作为GPIO使用，touchscreen驱动将GPIO81作为正常GPIO使用，将GPIO82当作中断使用。

## 确定需求

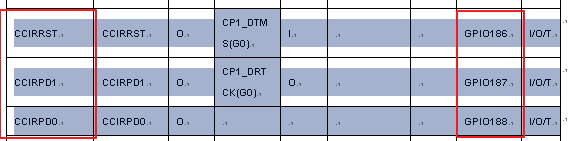
sprd\_sensor驱动中使用三个GPIO，分别是GPIO186，GPIO187，GPIO188，touchscreen驱动使用GPIO82作为touch中断。查看pinmap spec，spec路径如图中红框标注，找到这三个GPIO对应的pin name。

* + 1. 访问如下路径spec，

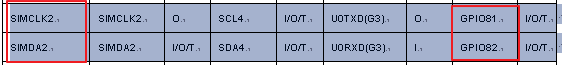


* + 1. 查看1.1.3节，pin multiplexed function list，找到GPIO对应pin name，图中红框标出分别为pin name 和对应的GPIO号。

范例一：sprd\_sensor GPIO



范例二：touchscreen irq



## 复用配置

在u-boot中配置pinmap复用关系，文件路径u-boot/board/spreadtrum/${BOARD}/pinmap.c.注意这里是根据pin name进行配置。三个GPIO分别对应的pin name为REG\_PIN\_CCIRRST，REG\_PIN\_CCIRPD1，REG\_PIN\_CCIRPD0，FUNC3对应pin的复用功能为GPIO。

|  |
| --- |
| struct other\_pin\_ctr\_reg other**[]** **=** **{**  ……  **/\***  **\* sprd\_sensor GPIO**  **\*/**  /\* reg |pin pull up|function|func pull up|sleep pull up|drv level|sleep i/o enable|schmitt enable|sleep enable \*/  **{**REG\_PIN\_CCIRRST**,** PIN\_NULL**,** FUNC3**,** PIN\_NULL**,** PIN\_NULL**,** DS\_L1**,** PIN\_NULL**,** PIN\_NULL**,** PIN\_NULL**},**  **{**REG\_PIN\_CCIRPD1**,** PIN\_NULL**,** FUNC3**,** PIN\_NULL**,** PIN\_NULL**,** DS\_L1**,** PIN\_NULL**,** PIN\_NULL**,** PIN\_NULL**},**  **{**REG\_PIN\_CCIRPD0**,** PIN\_NULL**,** FUNC3**,** PIN\_NULL**,** PIN\_NULL**,** DS\_L1**,** PIN\_NULL**,** PIN\_NULL**,** PIN\_NULL**},**  ……  **/\***  **\* touchscreen irq**  **\*/**  {REG\_PIN\_SIMCLK2, PIN\_NULL, FUNC3, FUNC\_WPU, SLP\_WPU, DS\_L1, SLP\_OE, PIN\_NULL, PIN\_NULL},  {REG\_PIN\_SIMDA2, PIN\_NULL, FUNC3, FUNC\_WPU, SLP\_WPU, DS\_L1, SLP\_OE, PIN\_NULL, PIN\_NULL},  **};** |

## 功能实现

1. GPIO功能

|  |
| --- |
| #define GPIO\_TOUCH\_RESET 81  #define GPIO\_TOUCH\_IRQ 82  #define GPIO\_SENSOR\_RESET 186//41  #define GPIO\_MAIN\_SENSOR\_PWN 187//42  #define GPIO\_SUB\_SENSOR\_PWN 188//43 |

1. Sprd sensor驱动中GPIO使用

|  |
| --- |
| **/\***  **\*probe函数中获取GPIO号，并且申请GPIO**  **\*/**  int **sensor\_k\_probe(**struct platform\_device **\***pdev**)**  **{**  ……  **/\***  **\*从device tree中或者是platform data中获取GPIO号码**  **\*/**  #ifdef CONFIG\_OF  sensor\_dev**.**this\_device**->**of\_node **=** pdev**->**dev**.**of\_node**;**  s\_p\_sensor\_mod**->**pin\_reset **=** of\_get\_gpio**(**sensor\_dev**.**this\_device**->**of\_node**,**0**);**  s\_p\_sensor\_mod**->**pin\_reset\_sub **=** s\_p\_sensor\_mod**->**pin\_reset**;**  s\_p\_sensor\_mod**->**pin\_main **=** of\_get\_gpio**(**sensor\_dev**.**this\_device**->**of\_node**,**1**);**  s\_p\_sensor\_mod**->**pin\_sub **=** of\_get\_gpio**(**sensor\_dev**.**this\_device**->**of\_node**,**2**);**  #else  s\_p\_sensor\_mod**->**pin\_reset **=** GPIO\_SENSOR\_RESET**;**  s\_p\_sensor\_mod**->**pin\_reset\_sub **=** GPIO\_SUB\_SENSOR\_RESET**;**  s\_p\_sensor\_mod**->**pin\_main**=** GPIO\_MAIN\_SENSOR\_PWN**;**  s\_p\_sensor\_mod**->**pin\_sub**=** GPIO\_SUB\_SENSOR\_PWN**;**  #endif  ……  **/\***  **\*根据GPIO号码申请GPIO**  **\*/**  ret **=** gpio\_request**(**s\_p\_sensor\_mod**->**pin\_main**,** "main camera"**);**  **if** **(**ret**)** **{**  tmp **=** s\_p\_sensor\_mod**->**pin\_main**;**  **goto** gpio\_err\_exit**;**  **}**  ret **=** gpio\_request**(**s\_p\_sensor\_mod**->**pin\_sub**,** "sub camera"**);**  **if** **(**ret**)** **{**  tmp **=** s\_p\_sensor\_mod**->**pin\_sub**;**  **goto** gpio\_err\_exit**;**  **}**  ret **=** gpio\_request**(**s\_p\_sensor\_mod**->**pin\_reset**,** "ccirrst"**);**  **if** **(**ret**)** **{**  tmp **=** s\_p\_sensor\_mod**->**pin\_reset**;**  **goto** gpio\_err\_exit**;**  **}**  **}**  **/\***  **\*remove函数中释放GPIO**  **\*/**  LOCAL int **sensor\_k\_remove(**struct platform\_device **\***dev**)**  **{**  printk**(**KERN\_INFO "sensor remove called !\n"**);**  **/\***  **\*remove函数中释放GPIO**  **\*/**  gpio\_free**(**s\_p\_sensor\_mod**->**pin\_reset**);**  gpio\_free**(**s\_p\_sensor\_mod**->**pin\_sub**);**  gpio\_free**(**s\_p\_sensor\_mod**->**pin\_main**);**  misc\_deregister**(&**sensor\_dev**);**  printk**(**KERN\_INFO "sensor remove Success !\n"**);**  **return** 0**;**  **}**  **/\***  **\*设置GPIO输入输出，实现模块控制**  **\*/**  LOCAL int **\_sensor\_k\_powerdown(**BOOLEAN power\_level**)**  **{**  **switch** **(**\_sensor\_K\_get\_curId**())** **{**  **case** SENSOR\_MAIN**:**  **{**  SENSOR\_PRINT\_HIGH**(**"SENSOR: pwdn %d,pin\_main %d\n"**,** power\_level**,**s\_p\_sensor\_mod**->**pin\_main**);**  **if** **(**0 **==** power\_level**)** **{**  **/\***  **\*设置GPIO为输出，输出低电平**  **\*/**  gpio\_direction\_output**(**s\_p\_sensor\_mod**->**pin\_main**,** 0**);**  **}** **else** **{**  **/\***  **\*设置GPIO为输出，输出高电平**  **\*/**  gpio\_direction\_output**(**s\_p\_sensor\_mod**->**pin\_main**,** 1**);**  **}**  **break;**  **}**  **case** SENSOR\_SUB**:**  **{**  SENSOR\_PRINT\_HIGH**(**"SENSOR: pwdn %d,pin\_sub %d\n"**,** power\_level**,**s\_p\_sensor\_mod**->**pin\_sub**);**  **if** **(**0 **==** power\_level**)** **{**  **/\***  **\*设置GPIO为输出，输出低电平**  **\*/**  gpio\_direction\_output**(**s\_p\_sensor\_mod**->**pin\_sub**,** 0**);**  **}** **else** **{**  **/\***  **\*设置GPIO为输出，输出低电平**  **\*/**  gpio\_direction\_output**(**s\_p\_sensor\_mod**->**pin\_sub**,** 1**);**  **}**  **break;**  **}**  **default:**  **break;**  **}**  **return** SENSOR\_K\_SUCCESS**;**  **}** |

1. Touchscreen驱动中GPIO使用

|  |
| --- |
| static void ft5x0x\_ts\_hw\_init**(**struct ft5x0x\_ts\_data **\***ft5x0x\_ts**)**  **{**  struct regulator **\***reg\_vdd**;**  struct i2c\_client **\***client **=** ft5x0x\_ts**->**client**;**  struct ft5x0x\_ts\_platform\_data **\***pdata **=** ft5x0x\_ts**->**platform\_data**;**  ……  **/\*申请GPIO，并设置输入输出方向\*/**  gpio\_request**(**pdata**->**irq\_gpio\_number**,** "ts\_irq\_pin"**);**  gpio\_request**(**pdata**->**reset\_gpio\_number**,** "ts\_rst\_pin"**);**  gpio\_direction\_output**(**pdata**->**reset\_gpio\_number**,** 1**);**  gpio\_direction\_input**(**pdata**->**irq\_gpio\_number**);/\*irq一定要作为输入\*/**  ……  **}**  **/\***  **\*probe函数中获取GPIO号，并且申请GPIO/IRQ**  **\*/**  static int ft5x0x\_ts\_probe**(**struct i2c\_client **\***client**,** const struct i2c\_device\_id **\***id**)**  **{**  **……**  ft5x0x\_ts\_hw\_init**(**ft5x0x\_ts**);/\*上面有代码实现\*/**  i2c\_set\_clientdata**(**client**,** ft5x0x\_ts**);**  client**->**irq **=** gpio\_to\_irq**(**pdata**->**irq\_gpio\_number**);/\*把GPIO号转换成irq号\*/**  **……**  **/\*申请irq号中断\*/**  #if USE\_THREADED\_IRQ  err **=** request\_threaded\_irq**(**client**->**irq**,** **NULL,** ft5x0x\_ts\_interrupt**,**  IRQF\_TRIGGER\_FALLING **|** IRQF\_ONESHOT**,** client**->**name**,** ft5x0x\_ts**);**  #else  err **=** request\_irq**(**client**->**irq**,** ft5x0x\_ts\_interrupt**,**  IRQF\_TRIGGER\_FALLING **|** IRQF\_ONESHOT**,** client**->**name**,** ft5x0x\_ts**);**  #endif  **……**  **}** |