# [Linux SPI总线和设备驱动架构](http://blog.csdn.net/droidphone/article/details/23367051)

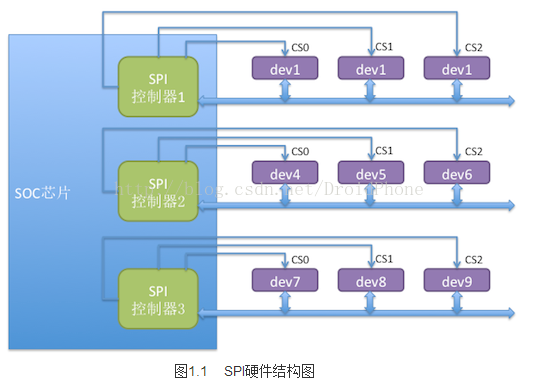
SPI是"Serial Peripheral Interface" 的缩写，是一种四线制的同步串行通信接口，用来连接微控制器、传感器、存储设备，SPI设备分为主设备和从设备两种，用于通信和控制的四根线分别是：

* CS    片选信号
* SCK  时钟信号
* MISO  主设备的数据输入、从设备的数据输出脚
* MOSI  主设备的数据输出、从设备的数据输入脚

因为在大多数情况下，CPU或SOC一侧通常都是工作在主设备模式，所以，目前的Linux内核版本中，只实现了主模式的驱动框架。

## 硬件结构

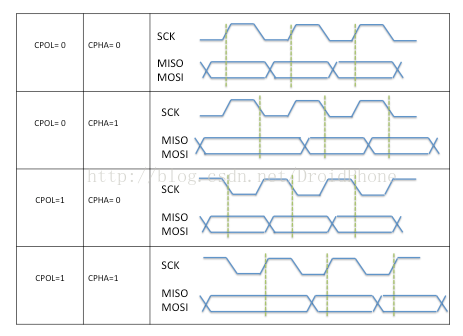
通常，负责发出时钟信号的设备我们称之为主设备，另一方则作为从设备，下图是一个SPI系统的硬件连接示例：



如上图所示，主设备对应SOC芯片中的SPI控制器，通常，一个SOC中可能存在多个SPI控制器，像上面的例子所示，SOC芯片中有3个SPI控制器。每个控制器下可以连接多个SPI从设备，每个从设备有各自独立的CS引脚。每个从设备共享另外3个信号引脚：SCK、MISO、MOSI。任何时刻，只有一个CS引脚处于有效状态，与该有效CS引脚连接的设备此时可以与主设备（SPI控制器）通信，其它的从设备处于等待状态，并且它们的3个引脚必须处于高阻状态。

## 工作时序

按照时钟信号和数据信号之间的相位关系，SPI有4种工作时序模式：



我们用CPOL表示时钟信号的初始电平的状态，CPOL为0表示时钟信号初始状态为低电平，为1表示时钟信号的初始电平是高电平。另外，我们用CPHA来表示在那个时钟沿采样数据，CPHA为0表示在首个时钟变化沿采样数据，而CPHA为1则表示要在第二个时钟变化沿来采样数据。内核用CPOL和CPHA的组合来表示当前SPI需要的工作模式：

* CPOL＝0，CPHA＝1        模式0
* CPOL＝0，CPHA＝1        模式1
* CPOL＝1，CPHA＝0        模式2
* CPOL＝1，CPHA＝1        模式3

## Linux SPI 驱动总体框架

在2.6的linux内核中，SPI的驱动架构可以分为如下三个层次：SPI 核心层、SPI控制器驱动层和SPI设备驱动层。Linux 中SPI驱动代码位于drivers/spi目录。

### SPI核心层

SPI核心层是Linux的SPI核心部分，提供了核心数据结构的定义、SPI控制器驱动和设备驱动的注册、注销管理等API。其为硬件平台无关层，向下屏蔽了物理总线控制器的差异，定义了统一的访问策略和接口；其向上提供了统一的接口，以便SPI设备驱动通过总线控制器进行数据收发。

Linux中，SPI核心层的代码位于driver/spi/ spi.c。由于该层是平台无关层，本文将不再叙述，有兴趣可以查阅相关资料。

### SPI控制器驱动层

      SPI控制器驱动层，每种处理器平台都有自己的控制器驱动，属于平台移植相关层。它的职责是为系统中每条SPI总线实现相应的读写方法。在物理上，每个SPI控制器可以连接若干个SPI从设备。

      在系统开机时，SPI控制器驱动被首先装载。一个控制器驱动用于支持一条特定的SPI总线的读写。一个控制器驱动可以用数据结构struct spi\_master来描述。在include/liunx/spi/spi.h文件中定义

### SPI设备驱动层

SPI设备驱动层为用户接口层，其为用户提供了通过SPI总线访问具体设备的接口。

 SPI设备驱动层可以用两个模块来描述，struct spi\_driver和struct spi\_device。在include/liunx/spi/spi.h文件中中定义