SPI 接口　简单不简单

SPI 串行外设接口总线，最早由 Motorola 提出，出现在其 M68 系列单片机中，由于其简单实用，又不牵涉到专利问题，因此许多厂家的设备都支持该接口，广泛应用于外设控制领域。

SPI 接口是一种事实标准，并没有标准协议，大部分厂家都是参照 Motorola 的 SPI 接口定义来设计的。但正因为没有确切的版本协议，不同家产品的 SPI 接口在技术上存在一定的差别，容易引起歧义，有的甚至无法直接互连（需要软件进行必要的修改）。

虽然 SPI 接口的内容非常简单，但本文仍将就其中的一些容易忽视的问题进行讨论。

SPI (Serial Peripheral Interface)

SPI 接口是 Motorola 首先提出的全双工三线同步串行外围接口，采用主从模式（Master Slave）架构；支持多 slave 模式应用，一般仅支持单 Master。

时钟由 Master 控制，在时钟移位脉冲下，数据按位传输，高位在前，低位在后（MSB first）；SPI 接口有 2 根单向数据线，为全双工通信，目前应用中的数据速率可达几 Mbps 的水平。

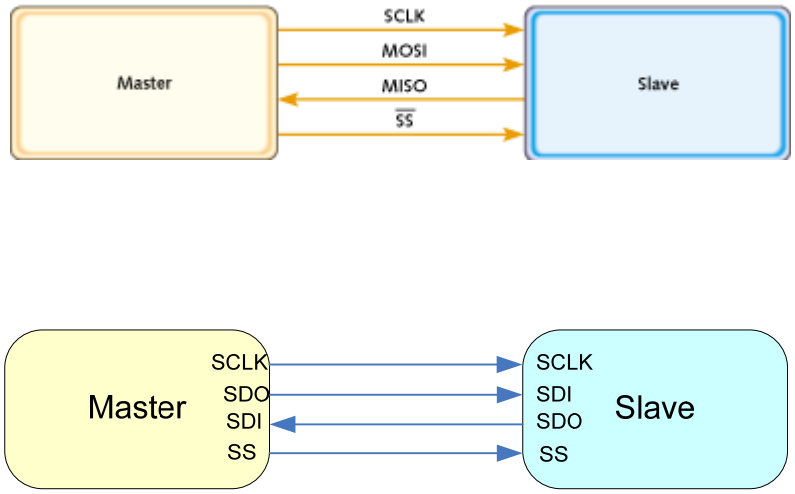

图 1　SPI 接口连接示意图

SPI 接口信号线

SPI 接口共有 4 根信号线，分别是：**设备选择线、时钟线、串行输出数据线、串行输入数据线**。

- **设备选择线 SS-（Slave select，或 CS-）**

SS-线用于选择激活某 Slave 设备，低有效，由 Master 驱动输出。只有当 SS-信号线为低电平时，对应 Slave 设备的 SPI 接口才处于工作状态。

- **SCLK：同步时钟信号线，**

SCLK 用来同步主从设备的数据传输，由 Master 驱动输出，Slave 设备按 SCK 的步调接收或发送数据。

- **串行数据线：**

SPI 接口数据线是单向的，共有两根数据线，分别承担 Master 到 Slave、Slave 到 Master 的数据传输；但是不同厂家的数据线命名有差别。

Motorola 的经典命名是 MOSI 和 MISO，这是站在信号线的角度来命名的。

MOSI：When master, out line; when slave, in line

MISO：When master, in line; when slave, out line

比如 MOSI，该线上数据一定是 Master 流向 Slave 的。因此在电路板上，Master 的 MOSI 引脚应与 Slave 的 MOSI 引脚连接在一起。双方的 MISO 也应该连在一起，而不是一方的 MOSI 连接另一方的 MISO。

不过，也有一些产家（比如 Microchip）是按照类似 SDI,SDO 的方式来命名，这是站在器件的角度根据数据流向来定义的。

SDI：串行数据输入

SDO：串行数据输出

这种情况下，当 Master 与 Slave 连接时，就应该用一方的 SDO 连接另一个方的 SDI。

由于 SPI 接口数据线是单向的，故电路设计时，数据线连接一定要正确，必然是一方的输出连接另一方的输入。

其实这个问题本来很简单的，但由于不同厂家产品的命名习惯可能不同，因此还需小心，以免低级出错。

数据传输的时序模式

为了适用不同产品接口应用需要，SPI 接口定义了多种时序传输模式，并可通过设置接口单元寄存器中的相关控制位来选择。在 Motorola 的产品中，时序即是由两个控制位（极性控制、相位控制）来控制的。

时钟极性选择位 CPOL：

在设备被使能激活后，还未进行数据传输时或两个字节数据传输间歇期间（见图 3 中的①与②处），SCLK 处于空闲（Idle）电平，通过“CPOL 空闲状态极性控制位”可以选择此空闲电平电平是 0 还是 1。

时钟相位选择位 CPHA：

该控制位用来选择数据接收端设备的采样时刻。可能在 Idle to Active 的跳变沿（见图 3 中的红色圈处），也可能在 Active to Idle 的跳变沿（见图 3 中的蓝色圈处）。在该采样时刻，线上数据必须已经稳定可靠，因此数据发送端设备应提前将数据移出到数据线上。为了降低设计难度，大部分接口电路都是用同一时钟周期中前一个时钟沿（即相反时钟变化方向）将数据移出。

SPI 线上的 Master，Slave 设备必须根据具体情况设置匹配的传输时序模式，时序只有匹配，数据传输才能正常进行。如果设置的不匹配，可能导致数据接收方和发送方

在同一时钟沿作用，导致数据传输失败。

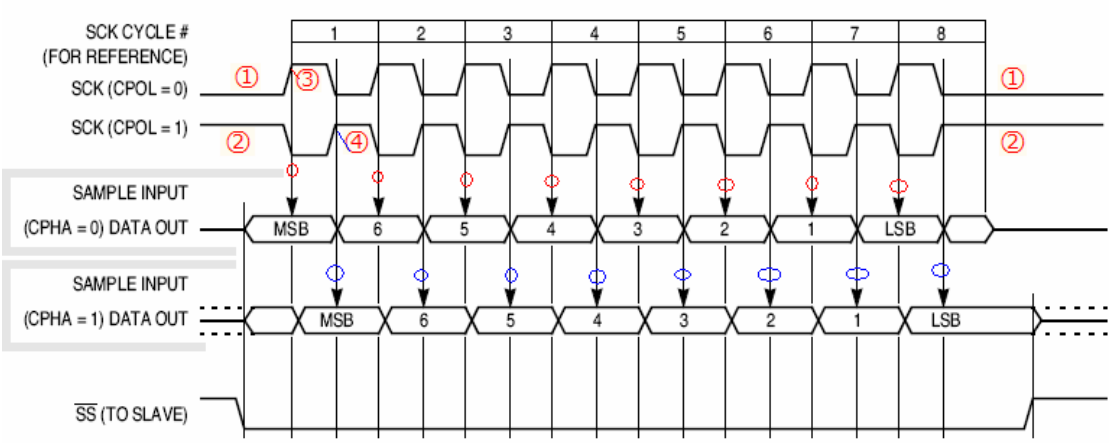

图 2 SPI 传输的几种时序模式

我们以手机设计中非常流行的触摸屏控制器 TSC2046 为例，介绍 SPI 接口的实际应用。

由于 TSC2046 采样触摸屏信息并量化出最高位需要一定时间，而 SPI 总线没有握手机制，为避免 Master 过早的启动传输，接收无效数据，TSC2046 引入了 BUSY 信号作为 TSC2046 向 Master 的指示。

TSC2046 是在时钟的第一个 Idle to Active 沿采集数据（下图 1 处），而在第一个（下一字节）Active to Idle 沿开始移出数据（下图 2 处），这导致 Master 只能在第二个 dle to Active 沿采集到的数据才是有效的（下图 3 处），而在第一个 Idle to Active 沿（下图 1 处）采集的数据是无效的，因此在软件中需要将该 Bit 丢弃。

可见，必须根据 Master 与 Slave 的实际时序进行匹配，软件也需要进行对应的调整，才能保证数据传输的正常进行。

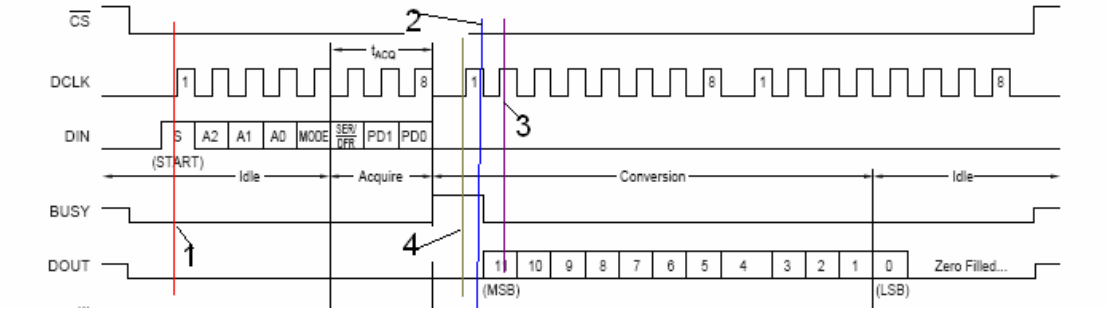

图 3 TSC2046 的 SPI 接口时序

多 Slave 的应用

SPI 也支持多 Slave 应用。多个 Slave 共享时钟线、数据线，可以直接并接在一起；而各 Slave 的片选线 SS 则单独与 Master 连接，受 Master 控制。在一段时间内，Master 只能通过某根 SS 线激活一个 Slave，进行数据传输，而此时其他 Slave 的时钟线和数据线端口则都应保持高阻状态，以免影响当前数据传输的进行。

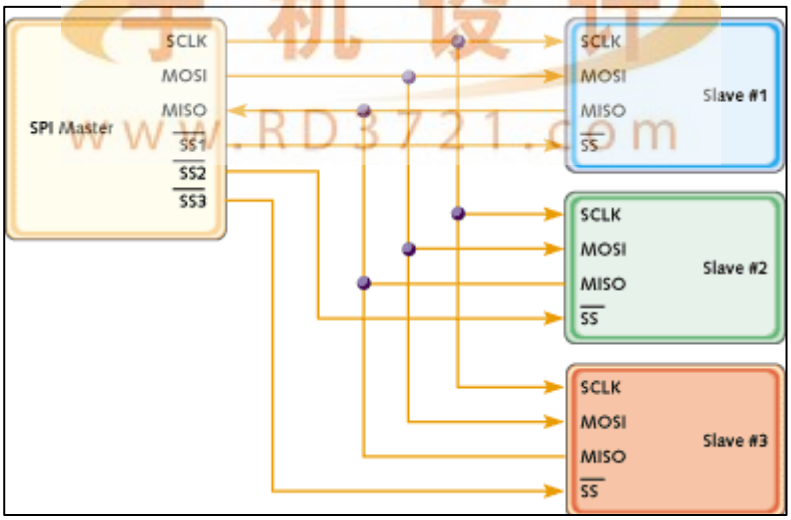

图 4　多 Slave 的 SPI 应用

SPI Vs I2C

SPI 协议没有定义寻址机制，需通过外部 SS 信号线选择设备，当出现多 slave 应用时，需要多根 SS 信号线，实施起来较 I2C 要复杂。此外，SPI 总线不支持总线控制权仲裁，故只能用在单 Master 的场合；而 I2C 可以支持多 Master 的应用。

SPI 协议相对 I2C 要简单，没有握手机制，数据传输效率高，速率也更快，通常应用中可达几 Mbps；此外 SPI 是全双工通信，可同时发送和接收数据，因此，SPI 比较适合用于数据传输的场合。比如需要较大批量数据传输的场合（比如 MMC/SD 卡的数据传输就支持 SPI 模式），或者无需寻址传输的场合。

而 I2C 协议功能较丰富，但也相对复杂，多用在传输一些控制命令字等有意义数据的场合。

比如 TSC2046 只有一个控制寄存器（一个 8bit 的命令字），使用 SPI 接口即可控制，因为无需寻址。而 OV 的 Cmos Sensor 内有多个控制寄存器，此时就必须使用 I2C 接口才能实现寻址控制（哦，确切的是 SCCB，一个很像 I2C 的东东）。

SPI 接口属于一种非常基本的外设接口，但是应用却很广泛。SPI 也有所发展，比兔 NS 推出的 SPI 的精简接口 Microwire，满足通常外设的扩展需求。Motorola 还推出了扩展功能的 QSPI（Queued SPI）接口，应用更为广泛。

欢迎您对本文作出评价，特别是对其中的问题和错误作出指正，请与 RD3721 编辑联系(edit@rd3721.com)!