一、概述.

SPI, Serial Perripheral Interface, 串行外围设备接口, 是 Motorola 公司推出的一种同步串行接口技术. SPI 总线在物理上是通过接在外围设备微控制器(PICmicro) 上面的微处理控制单元 (MCU) 上叫作同步串行端口(Synchronous Serial Port) 的模块(Module)来实现的, 它允许 MCU 以全双工的同步串行方式, 与各种外围设备进行高速数据通信.

SPI 主要应用在 EEPROM, Flash, 实时时钟(RTC), 数模转换器(ADC), 数字信号处理器(DSP) 以及数字信号解码器之间. 它在芯片中只占用四根管脚 (Pin) 用来控制以及数据传输, 节约了芯片的 pin 数目, 同时为 PCB 在布局上节省了空间. 正是出于这种简单易用的特性, 现在越来越多的芯片上都集成了 SPI 技术.

二、 特点

1. 采用主-从模式(Master-Slave) 的控制方式

SPI 规定了两个 SPI 设备之间通信必须由主设备 (Master) 来控制次设备 (Slave). 一个 Master 设备可以通过提供 Clock 以及对 Slave 设备进行片选 (Slave Select) 来控制多个 Slave 设备, SPI 协议还规定 Slave 设备的 Clock 由 Master 设备通过 SCK 管脚提供给 Slave 设备, Slave 设备本身不能产生或控制 Clock, 没有 Clock 则 Slave 设备不能正常工作.

2. 采用同步方式(Synchronous)传输数据

Master 设备会根据将要交换的数据来产生相应的时钟脉冲(Clock Pulse), 时钟脉冲组成了时钟信号(Clock Signal) , 时钟信号通过时钟极性 (CPOL) 和 时钟相位 (CPHA) 控制着两个 SPI 设备间何时数据交换以及何时对接收到的数据进行采样, 来保证数据在两个设备之间是同步传输的.

3. 数据交换(Data Exchanges)

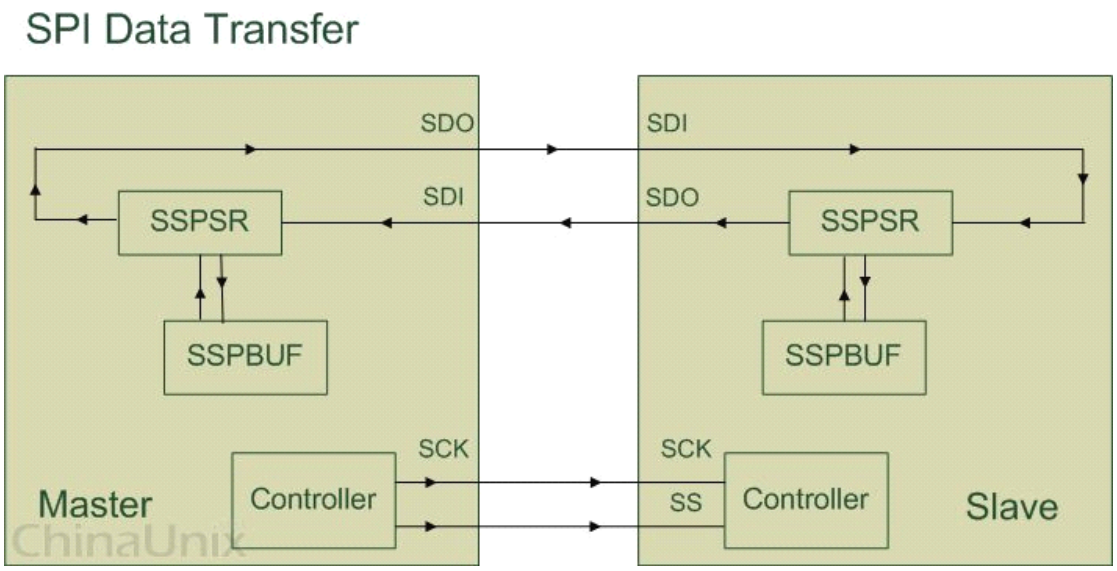
SPI 设备间的数据传输之所以又被称为数据交换, 是因为 SPI 协议规定一个 SPI 设备不能在数据通信过程中仅仅只充当一个 "发送者(Transmitter)" 或者 "接收者 (Receiver)". 在每个 Clock 周期内, SPI 设备都会发送并接收一个 bit 大小的数据, 相当于该设备有一个 bit 大小的数据被交换了.

一个 Slave 设备要想能够接收到 Master 发过来的控制信号, 必须在此之前能够被 Master 设备进行访问 (Access). 所以, Master 设备必须首先通过 SS/CS pin 对 Slave 设备进行片选, 把想要访问的 Slave 设备选上.

在数据传输的过程中, 每次接收到的数据必须在下一次数据传输之前被采样. 如果之前接收到的数据没有被读取, 那么这些已经接收完成的数据将有可能会被丢弃, 导致 SPI 物理模块最终失效. 因此, 在程序中一般都会在 SPI 传输完数据后, 去读取 SPI 设备里的数据, 即使这些数据(Dummy Data)在我们的程序里是无用的.

三、 工作机制

1. 概述



上图只是对 SPI 设备间通信的一个简单的描述，下面就来解释一下图中所示的几个组件(Module):

SSPBUF, Synchronous Serial Port Buffer, 泛指 SPI 设备里面的内部缓冲区，一般在物理上是以 FIFO 的形式，保存传输过程中的临时数据;

SSPSR, Synchronous Serial Port Register, 泛指 SPI 设备里面的移位寄存器(Shift Regitser), 它的作用是根据设置好的数据位宽(bit-width) 把数据移入或者移出 SSPBUF;

Controller, 泛指 SPI 设备里面的控制寄存器，可以通过配置它们来设置 SPI 总线的传输模式.

通常情况下，我们只需要对上图所描述的四个管脚(pin) 进行编程即可控制整个 SPI 设备之间的数据通信:

SCK, Serial Clock, 主要的作用是 Master 设备往 Slave 设备传输时钟信号，控制数据交换的时机以及速率;

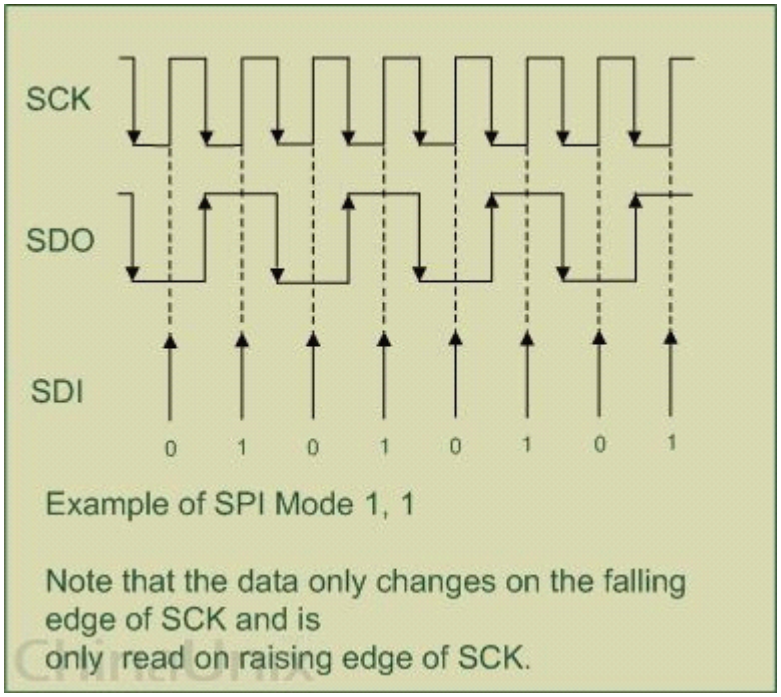
SS/CS, Slave Select/Chip Select, 用于 Master 设备片选 Slave 设备，使被选中的 Slave 设备能够被 Master 设备所访问;

SDO/MOSI, Serial Data Output/Master Out Slave In, 在 Master 上面也被称为 Tx-Channel, 作为数据的出口，主要用于 SPI 设备发送数据;

SDI/MISO, Serial Data Input/Master In Slave Out, 在 Master 上面也被称为 Rx-Channel, 作为数据的入口，主要用于 SPI 设备接收数据;

SPI 设备在进行通信的过程中, Master 设备和 Slave 设备之间会产生一个数据链路回环(Data Loop), 就像上图所画的那样，通过 SDO 和 SDI 管脚, SSPSR 控制数据移入移出 SSPBUF, Controller 确定 SPI 总线的通信模式, SCK 传输时钟信号.

2. Timing.



上图通过 Master 设备与 Slave 设备之间交换 1 Byte 数据来说明 SPI 协议的工作机制.

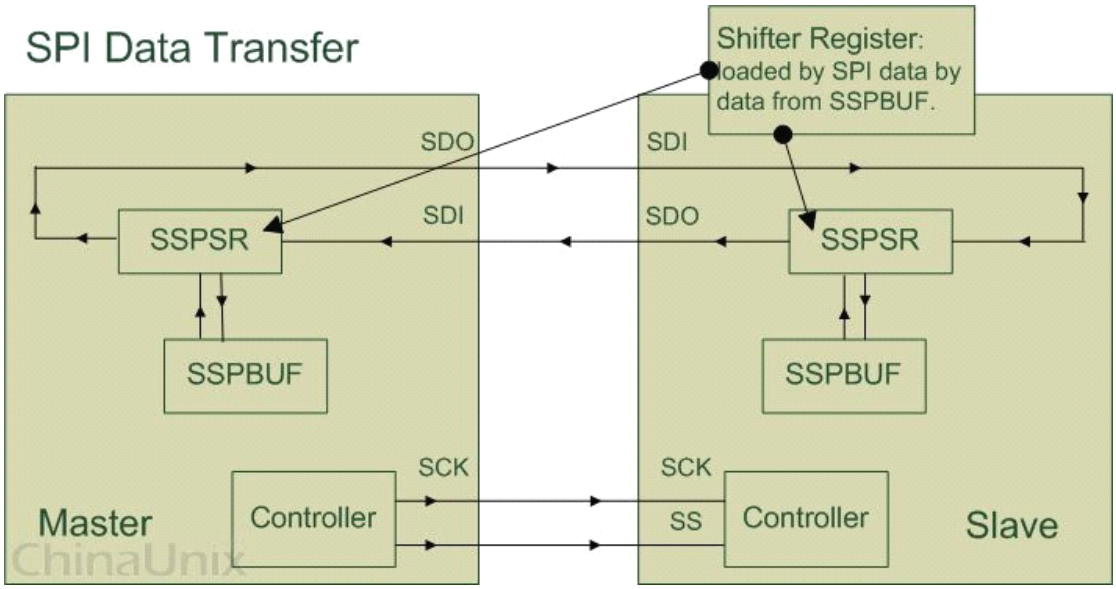
首先, 在这里解释一下两个概念:

CPOL: 时钟极性, 表示 SPI 在空闲时, 时钟信号是高电平还是低电平. 若 CPOL 被设为 1, 那么该设备在空闲时 SCK 管脚下的时钟信号为高电平. 当 CPOL 被设为 0 时则正好相反.

CPHA: 时钟相位, 表示 SPI 设备是在 SCK 管脚上的时钟信号变为上升沿时触发数据采样, 还是在时钟信号变为下降沿时触发数据采样. 若 CPHA 被设置为 1, 则 SPI 设备在时钟信号变为下降沿时触发数据采样, 在上升沿时发送数据. 当 CPHA 被设为 0 时也正好相反.

上图里的 "Mode 1, 1" 说明了本例所使用的 SPI 数据传输模式被设置成 CPOL = 1, CPHA = 1. 这样, 在一个 Clock 周期内, 每个单独的 SPI 设备都能以全双工 (Full-Duplex) 的方式, 同时发送和接收 1 bit 数据, 即相当于交换了 1 bit 大小的数据. 如果 SPI 总线的 Channel-Width 被设置成 Byte, 表示 SPI 总线上每次数据传输的最小单位为 Byte, 那么挂载在该 SPI 总线的设备每次数据传输的过程至少需要 8 个 Clock 周期(忽略设备的物理延迟). 因此, SPI 总线的频率越快, Clock 周期越短, 则 SPI 设备间数据交换的速率就越快.

3. SSPSR.



SSPSR 是 SPI 设备内部的移位寄存器(Shift Register). 它的主要作用是根据 SPI 时钟信号状态, 往 SSPBUF 里移入或者移出数据, 每次移动的数据大小由 Bus-Width 以及 Channel-Width 所决定.

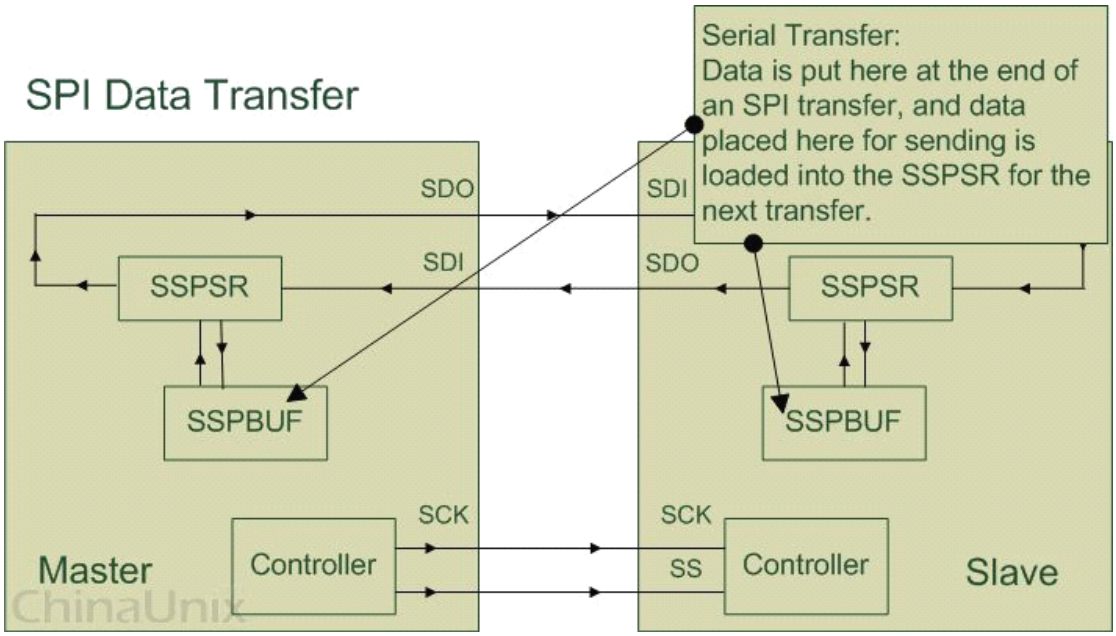
Bus-Width 的作用是指定地址总线到 Master 设备之间数据传输的单位.

例如, 我们想要往 Master 设备里面的 SSPBUF 写入 16 Byte 大小的数据: 首先, 给 Master 设备的配置寄存器设置 Bus-Width 为 Byte; 然后往 Master 设备的 Tx-Data 移位寄存器在地址总线的入口写入数据, 每次写入 1 Byte 大小的数据(使用 writeb 函数); 写完 1 Byte 数据之后, Master 设备里面的 Tx-Data 移位寄存器会自动把从地址总线传来的 1 Byte 数据移入 SSPBUF 里; 上述动作一共需要重复执行 16 次.

Channel-Width 的作用是指定 Master 设备与 Slave 设备之间数据传输的单位. 与 Bus-Width 相似, Master 设备内部的移位寄存器会依据 Channel-Width 自动地把数据从 Master-SSPBUF 里通过 Master-SDO 管脚搬运到 Slave 设备里的 Slave-SDI 引脚, Slave—SSPSR 再把每次接收的数据移入 Slave-SSPBUF 里.

通常情况下, Bus-Width 总是会大于或等于 Channel-Width, 这样能保证不会出现因 Master 与 Slave 之间数据交换的频率比地址总线与 Master 之间的数据交换频率要快, 导致 SSPBUF 里面存放的数据为无效数据这样的情况.

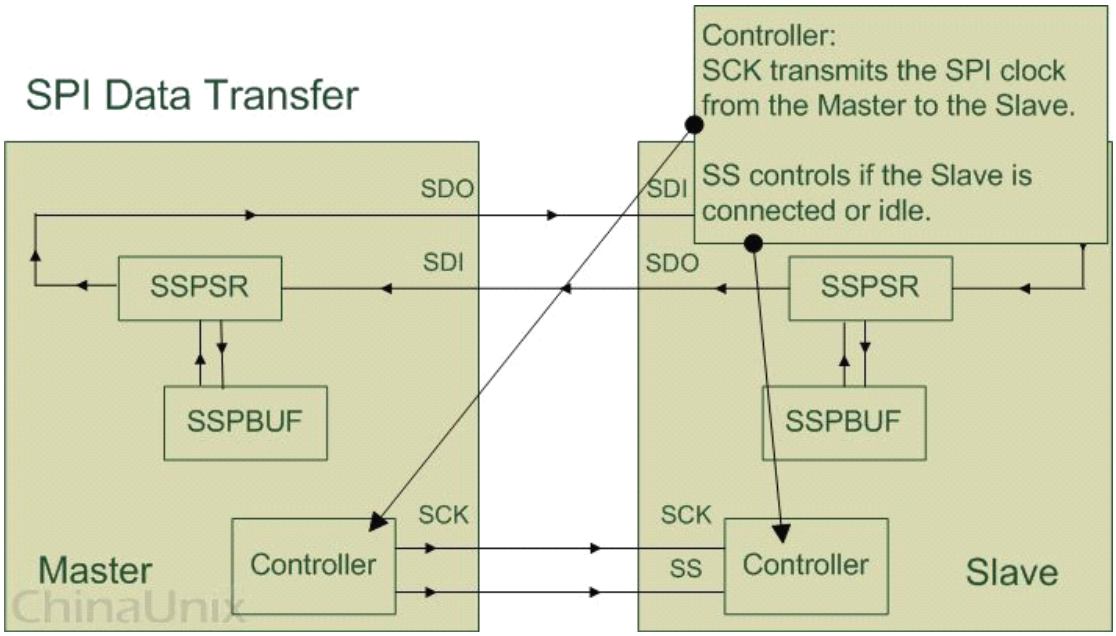
4. SSPBUF.



我们知道，在每个时钟周期内，Master 与 Slave 之间交换的数据其实都是 SPI 内部移位寄存器从 SSPBUF 里面拷贝的. 我们可以通过往 SSPBUF 对应的寄存器 (Tx-Data / Rx-Data register) 里读写数据，间接地操控 SPI 设备内部的 SSPBUF.

例如，在发送数据之前，我们应该先往 Master 的 Tx-Data 寄存器写入将要发送出去的数据，这些数据会被 Master-SSPSR 移位寄存器根据 Bus-Width 自动移入 Master-SSPBUF 里，然后这些数据又会被 Master-SSPSR 根据 Channel-Width 从 Master-SSPBUF 中移出，通过 Master-SDO 管脚传给 Slave-SDI 管脚，Slave-SSPSR 则把从 Slave-SDI 接收到的数据移入 Slave-SSPBUF 里. 与此同时，Slave-SSPBUF 里面的数据根据每次接收数据的大小(Channel-Width)，通过 Slave-SDO 发往 Master-SDI，Master-SSPSR 再把从 Master-SDI 接收的数据移入 Master-SSPBUF.在单次数据传输完成之后，用户程序可以通过从 Master 设备的 Rx-Data 寄存器读取 Master 设备数据交换得到的数据.

5. Controller.



Master 设备里面的 Controller 主要通过时钟信号(Clock Signal)以及片选信号(Slave Select Signal)来控制 Slave 设备. Slave 设备会一直等待, 直到接收到 Master 设备发过来的片选信号, 然后根据时钟信号来工作.

Master 设备的片选操作必须由程序所实现. 例如: 由程序把 SS/CS 管脚的时钟信号拉低电平, 完成 SPI 设备数据通信的前期工作; 当程序想让 SPI 设备结束数据通信时, 再把 SS/CS 管脚上的时钟信号拉高电平