



第11章 STC单片机SPI原理及实现

何宾

2015.02

本章主要内容

- STC SPI模块结构及功能
- SPI模块寄存器组
- SPI模块配置及时序
- SPI模块设计实例

STC SPI模块结构及功能

--SPI传输特点



使用SPI进行数据的传输，不但需要有源方（发送数据的一方）提供的时钟信号。而且，还需要有源方（发送数据的一方）提供的同步信号。

- 基于SPI的通信方式是典型的高速同步双向数据传输方式。
- 这种通信方式在工业界应用非常广泛。
 - 典型地，SPI接口主要用于在EEPROM、FLASH、实时时钟、AD转换器、数字信号处理器和数字信号解码器之间的数据传输。

STC SPI模块结构及功能

--STC单片机SPI模块功能

STC公司系列单片机提供了另一种高速串行通信接口——SPI接口。SPI接口为数据传输提供了主模式和从模式两种工作模式。

- 在主模式下，支持高达3MHzbps的数据传输率。如果单片机的主频在20~36MHz，工作频率为12MHz时，可以提供更高的工作速度。
- 在从模式下，速度受限，STC推荐数据率在SYSclk/4内的数据传输率。
- 此外，SPI接口提供了完成标志和写冲突标志保护。

STC SPI模块结构及功能

---SPI接口信号

在SPI接口中，提供了4个信号用于进行高速同步数据传输。包括：

MOSI

- 主设备输出和从设备输入信号，实现主设备（发出数据）到从设备（接收数据）的数据传输。
 - 当STC的SPI接口作为主设备传输数据时，该信号方向为输出，指向从设备；
 - 当STC的SPI接口作为从设备接收数据时，该信号方向为输入，由从设备指向STC单片机的SPI接口。

STC SPI模块结构及功能

---SPI接口信号

MISO

- 主设备输入和从设备输出信号，实现从设备（发出数据）到主设备（接收数据）的数据传输。
 - 当STC的SPI接口作为主设备传输数据时，该信号方向为输入，由从设备指向STC单片机的SPI接口；
 - 当STC的SPI接口作为从设备接收数据时，该信号方向为输出，指向从设备。

注：不管单片机的SPI接口是作为主设备还是从设备，MOSI和MISO传输方向都是相反的。

STC SPI模块结构及功能

---SPI接口信号

SCLK

■ 串行时钟信号，它由主设备发出，指向从设备。

- 在串行时钟的控制下，用于同步主设备和从设备之间MISO和MOSI信号线上数据的传输过程。
- 当主设备启动一次数据传输过程时，自动产生8个SCLK信号给从设备。
- 在SCLK信号的上升沿或者下降沿到来的时候，移出一位数据。
- 一次传输可以传输一个字节的的数据。

注：（1）在一些应用中，将多个设备SPI接口的SCLK、MOSI、MISO信号连接在一起。通过MOSI信号，将数据从主设备发送到从设备。

（2）如果将SPCTL寄存器的SPEN位设置为0（复位值为0），则禁止SPI接口，分配给这些信号的引脚可以当作普通I/O引脚使用。

STC SPI模块结构及功能

---SPI接口信号



SS

- 从设备选择信号。通过该信号，主设备用于选择处于从模式的SPI设备。在主模式和从模式下，SS信号的使用不同。
 - 在主模式下，SPI接口只能有一个主设备，不存在选择主机的问题，在该模式下，该位不是必须的。在主模式下，将主设备的SS引脚通过10KΩ电阻上拉到高电平。每一个从设备的SS信号与主设备的SS信号连接，由主设备控制电平的高低，以便主设备选择从设备。
 - 在从模式下，不管接收还是发送该信号必须有效。因此，在一次数据传输之前必须将SS信号拉低。

STC SPI模块结构及功能

---SPI接口信号

通过SS信号，SPI从设备确认是否被选中。如果满足下面的其中一个条件，则忽略该信号：

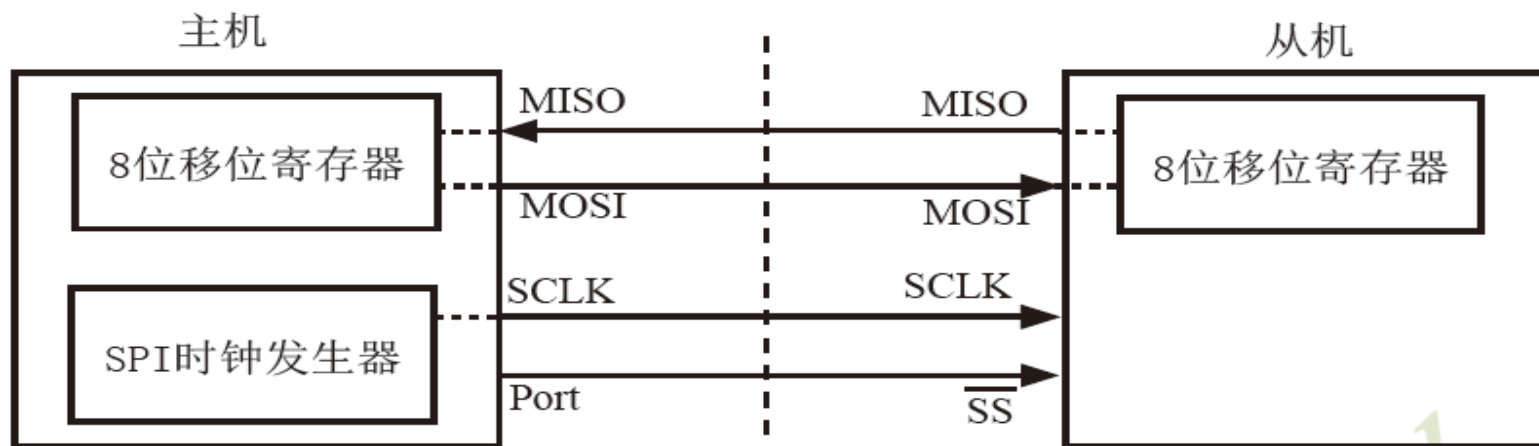
- 如果禁止SPI接口。
- 如果配置为SPI主设备，即SPCTL寄存器的MSTR位置为1，并且P1.2/SS配置为输出。
- 如果SPCTL寄存器的SSIG位置为高，该引脚用作普通I/O功能。

注：即使STC单片机的SPI接口配置为主设备，但是仍然可以通过拉低SS引脚，将其配置为从设备。通过设置寄存器相应的位使能该特性。

STC SPI模块结构及功能

--SPI接口的数据通信方式

STC15系列单片机的SPI接口提供的三种数据通信方式。
单一主设备和单一从设备方式



STC SPI模块结构及功能

--SPI接口的数据通信方式

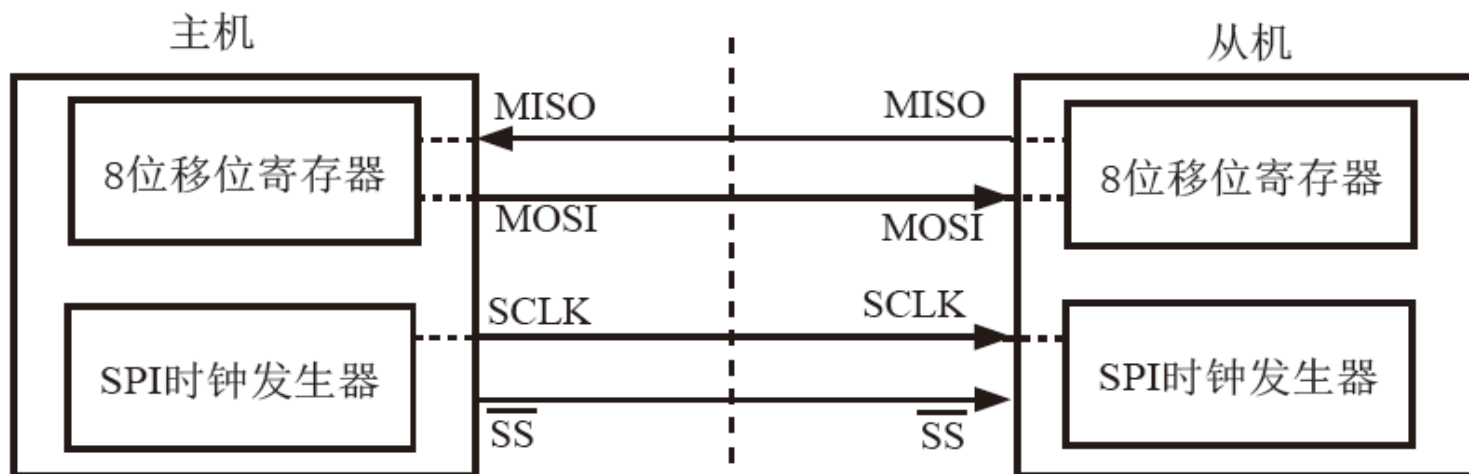
- 在这种通信配置模式中，从设备的SSIG位设置为0，SS用于选择从设备。
- SPI主设备可以使用任何引脚，包括P1.2/SS引脚来驱动SS信号。
- 主设备的SPI接口和从设备的SPI的8位移位寄存器构成一个循环的16位移位寄存器。
- 在该模式下，主设备既可以向从设备发送数据，又可以读取从设备发来的数据。

STC SPI模块结构及功能

--SPI接口的数据通信方式

双设备方式

- 设备可以互为主设备和从设备



STC SPI模块结构及功能

--SPI接口的数据通信方式

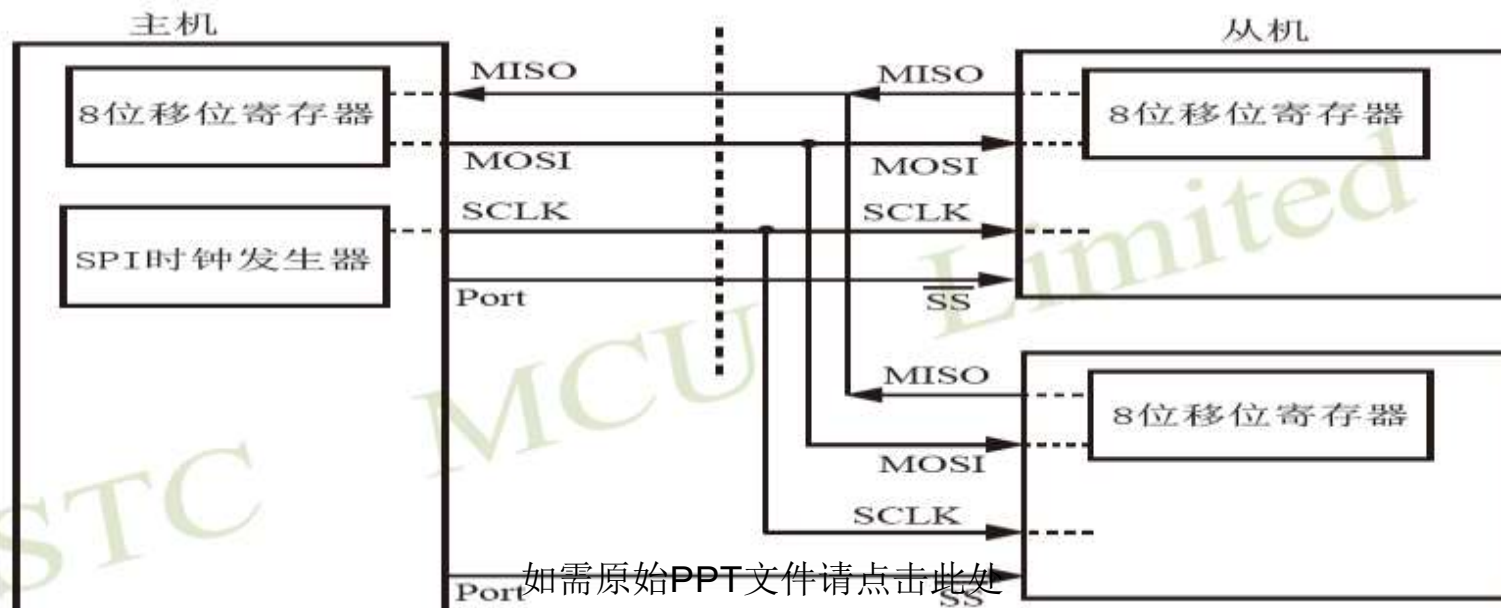
- 在该配置模式中，当没有SPI数据传输时，两个设备均可作为主机，将SSIG清零并将P1.2/SS引脚配置为准双向模式。
 - 当其中一个器件启动传输时，它将P1.2/SS配置为输出并驱动为低电平，这样就将另一个设备变成从设备。
- 双方初始化时，将自己配置成忽略SS引脚的从模式。
 - 当一方要主动发送数据时，先检测SS引脚的电平。如果SS引脚为高，就将自己设置为忽略SS引脚的主模式。
 - 在平时，通信双方将自己配置成没有选中的从模式。在该模式下，MISO、MOSI、SCLK信号均为输入。
 - 当多个单片机的SPI接口以该模式并联时不会发生总线冲突。

STC SPI模块结构及功能

--SPI接口的数据通信方式

单一主设备和多个从设备方式

- 在该配置中，从设备的SSIG位置为0，通过SS信号，选择对应的从设备。
- 主设备的SPI接口可以使用任何端口来驱动SS引脚。

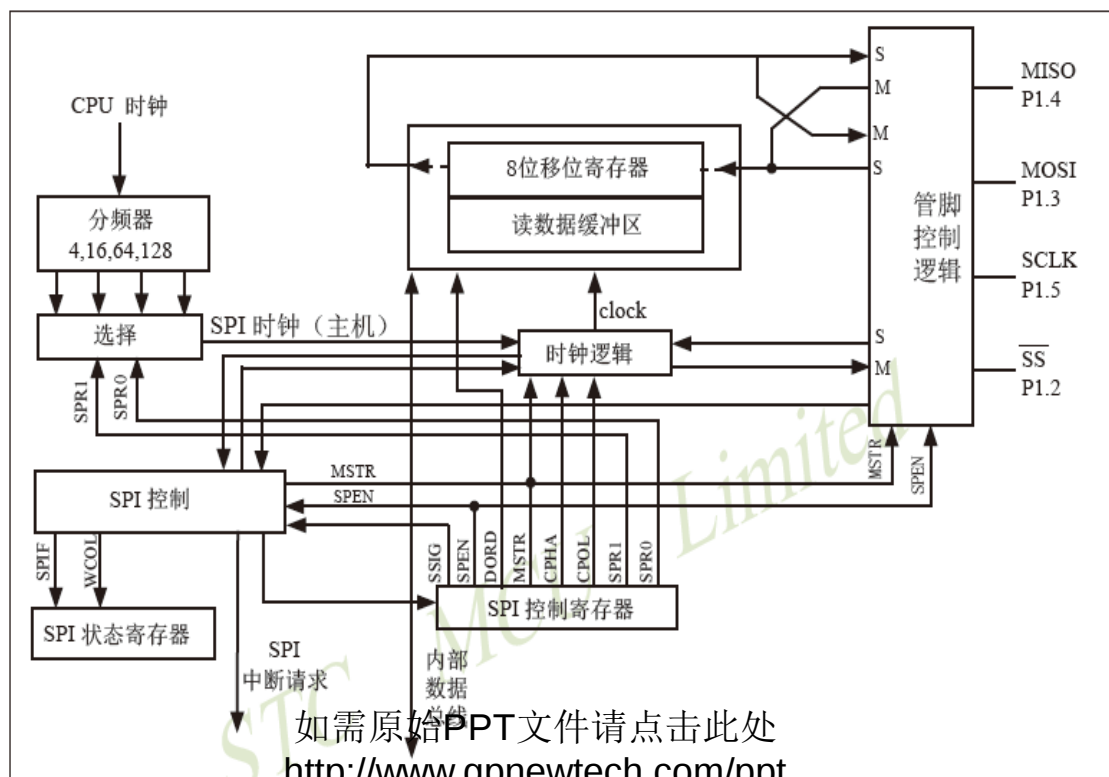


STC SPI模块结构及功能

--SPI模块内部结构

SPI模块核心是一个8位的移位寄存器和数据缓冲器，可以同时接收和发送数据。

■ 在数据传输的过程中，将接收和发送的数据保存在数据缓冲器。



STC SPI模块结构及功能

--SPI模块内部结构

- 对于主模式来说，如果要发送一个字节的数据，只需要将该数据写到SPDAT寄存器中。
 - 在该模式下，SS信号不是必需的；
- 在从模式下，必须在SS信号变为有效并接收到合适的时钟信号后，才可以开始进行数据传输。
 - 在从模式下，如果完成一个字节的数据传输，则SS信号变高，这个字节立刻被硬件逻辑标记为接收完成。随后，SPI接口准备接收下一个数据。

SPI模块寄存器组

--SPI控制寄存器

SPI控制寄存器SPCTL

- 该寄存器位于STC单片机特殊功能寄存器地址为0xCE的位置。
- 当复位后，该寄存器的值为“00000100”。

SPI控制寄存器SPCTL各位的含义

比特	B7	B6	B5	B4	B3	B2	B1	B0
名字	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

■ SSIG

- SS引脚忽略控制位。当该位为1时，MSTR位确定单片机是主设备还是从设备；当该位为0时，SS引脚用于确定单片机是主设备还是从设备。SS引脚可作为普通I/O。

SPI模块寄存器组

--SPI控制寄存器

■ SPEN

- SPI使能控制位。当该位为1时，使能SPI接口；当该位为0时，禁止SPI接口，此时所有SPI接口的信号引脚都可以作为普通I/O。

■ DORD

- 设定SPI数据发送和接收的位顺序。当该位为1时，先发送数据字的最低有效位（LSB）；当该位为0时，先发送数据字的最高有效位（MSB）。

■ MSTR

- 主从模式选择位。当该位为1时，主模式；当该位为0时，从模式。

SPI模块寄存器组

--SPI控制寄存器

■ CPOL : SPI时钟极性选择位。

- 当该位为1时，空闲情况下，SCLK为高电平。SCLK的前一个时钟沿为下降沿，而后一个时钟沿为上升沿。
- 当该位为0时，空闲情况下，SCLK为低电平。SCLK的前一个时钟沿为上升沿，而后一个时钟沿为下降沿。

■ CPHA : SPI时钟相位选择位。

- 当该位为1时，在SCLK的前时钟沿驱动数据，并在后时钟沿采样；
- 当该位为0时，在SS为低时驱动数据，在SCLK的后时钟沿改变数据，并在前时钟沿采样。

SPI模块寄存器组

--SPI控制寄存器组

■ SPR1和SPR0

□ 时钟速率选择位

SPR1和SPR0位的含义

SPR1	SPR0	时钟 (SCLK)
0	0	CPU_CLK/4
0	1	CPU_CLK/8
1	0	CPU_CLK/16
1	1	CPU_CLK/32

SPI模块寄存器组

--SPI状态寄存器

SPI状态寄存器SPSTAT

- 该寄存器位于STC单片机特殊功能寄存器地址为0xCD的位置。
- 当复位后，该寄存器的值为“00xxxxxx”。

SPI状态寄存器SPSTAT各位的含义

比特	B7	B6	B5	B4	B3	B2	B1	B0
名字	SPIF	WCOL	—	—	—	—	—	—

SPI模块寄存器组

--SPI状态寄存器

■ SPIF

- SPI传输完成标志。当完成一次SPI数据传输后，硬件将该位设置为1。此时，如果允许SPI中断，则产生中断。
- 当SPI处于主模式，且SSIG为0时，如果SS引脚为输入并驱动为低电平时，硬件也将该标志置为1，表示改变模式

■ WCOL

- SPI写冲突标志。在数据传输的过程中，如果对SPI数据寄存器SPDAT进行写操作，硬件将该标志置1。

SPI模块寄存器组

--SPI数据寄存器

SPI数据寄存器SPDAT

- 该寄存器位于STC单片机特殊功能寄存器地址为0xCF的位置。
- 当复位后，该寄存器的值为“00000000”。

SPI数据寄存器SPDAT各位的含义

比特	B7	B6	B5	B4	B3	B2	B1	B0
名字	8位数据							

SPI模块寄存器组

--中断优先级寄存器2

中断优先级寄存器2

- 该寄存器位于STC单片机特殊功能寄存器地址为0xB5的位置。
- 当复位后，该寄存器的值为“xxx00000”。

中断优先级控制寄存器IP2各位的含义

比特	B7	B6	B5	B4	B3	B2	B1	B0
名字	--	--	--	PX4	PPWMFD	PPWM	PSPI	PS2

■ PSPI

- SPI中断优先级控制位。当该位为0时，SPI中断为最低优先级中断（优先级为0）；当该位为1时，SPI中断为最高优先级中断（优先级1）。

SPI模块寄存器组

--中断允许寄存器2

中断允许寄存器IE2

- 该寄存器位于STC单片机特殊功能寄存器地址为0xAF的位置。
- 当复位后，该寄存器的值为“x000000”。

中断允许寄存器IE2各位的含义

比特	B7	B6	B5	B4	B3	B2	B1	B0
名字	--	ET4	ET3	ES4	ES3	ET2	ESPI	ES2

■ ESPI

- SPI中断允许位。当该位为1时，允许SPI中断；当该位为0时，禁止SPI中断。

SPI模块寄存器组

--控制SPI引脚位置寄存器

PCA模块引脚切换寄存器AUXR1 (P_SW1)

- 该寄存器位于STC单片机特殊功能寄存器地址为0xA2的位置。
- 当复位后，该寄存器的值为“00000000”。

PCA模块引脚切换寄存器AUXR1 (P_SW1) 各位的含义

比特	B7	B6	B5	B4	B3	B2	B1	B0
名字	S1_S1	S1_S0	CCP_S1	CCP_S0	SPI_S1	SPI_S0	0	DPS

SPI模块寄存器组

--控制SPI引脚位置寄存器

■ SPI_S1和SPI_S0确定SPI接口在单片机上引脚的位置

SPI_S1和SPI_S0各位的含义

SPI_S1	SPI_S0	功能
0	0	选择SPI接口分别对应于单片机P1.2/SS、P1.3/MOSI、P1.4/MISO、P1.5/SCLK引脚
0	1	选择SPI接口分别对应于单片机P2.4/SS_2、P2.3/MOSI_2、P2.2/MISO_2、P2.1/SCLK_2引脚
1	0	选择SPI接口分别对应于单片机P5.4/SS_3、P4.0/MOSI_3、P4.1/MISO_3、P4.3/SCLK_3引脚
1	1	无效

SPI模块配置及时序

--SPI配置模式

STC 15系列单片机进行SPI通信时，通过SPEN位、SSIG位、SS引脚和MSTR位，控制其工作模式

SPI模块配置及时序

--SPI配置模式

主从模式的选择

SPEN	SSIG	SS引脚 /P1.2	MSTR	主/从模式	MISO /P1.4	MOSI /P1.3	SCLK /P1.5	功能
0	X	P1.2/SS	X	禁止SPI	MISOP 1.4	MOSI/ P1.3	SCLK/ P1.5	禁止SPI
1	0	0	0	从模式	输出	输入	输入	选择作为从设备
1	0	1	0	从模式, 未选中	高阻	输入	输入	未被选中。MISO为高阻状态, 以避免总线冲突
1	0	0	1->0	从模式	输出	输入	输入	P1.2/SS配置为输入或准双向口, SSIG为0, 如果将SS驱动为低, 则选择作为从设备。当SS变为低电平时, 将MSTR清零

SPI模块配置及时序

--SPI配置模式

主从模式的选择

SPEN	SSIG	SS引脚 /P1.2	MSTR	主/从模 式	MISO /P1.4	MOSI /P1.3	SCLK /P1.5	功能
1	0	1	1	主（空闲）	输入	高阻	高阻	当主机空闲时，MOSI和SCLK为高阻，以避免总线冲突。用户必须将SCLK上拉或者下拉，以避免SCLK处于悬空状态
				主（活动）	输入	输出	输出	作为主机激活时，MOSI和SCLK为推挽输出
1	1	P1.2/SS	0	从模式	输出	输入	输入	
1	1	P1.2/SS	1	主模式	输入	输出	输出	

如需原始PPT文件请点击此处

<http://www.gpnewtech.com/ppt>

主/从模式的注意事项

--作为从设备时的注意事项

- 在SPI中，总是由主设备发起数据传输过程。
 - 如果使能SPI，并将其设置为主设备，主设备对SPI数据寄存器的写操作将启动SPI时钟发生器和数据的传输。在数据写入SPDAT之后的0.5~1个SPI比特位时间后，在MOSI引脚上将出现数据。
- 传输完一个字节后，停止SPI时钟，将SPIF标志置1，并产生一个中断。
- 主设备和从设备CPU的两个移位寄存器可以看作是一个16位的循环移位寄存器。
 - 当数据从主设备移位传输到从设备的同时，数据以反方向从从设备移位传输到主设备。也就是，在一个移位周期过程中，主设备和从设备相互交换数据。

主/从模式的注意事项

--作为主设备时的注意事项

■ 当CPHA为0时，SSIG必须为0

- 不能忽略SS引脚，SS引脚必须设置为低，并且在每个连续的串行字节发送完后必须重新设置为高电平。
- 如果在SS低电平有效时，执行对SPDAT寄存器的写操作，将会导致出现一个写冲突错误。

■ 当CPHA为1时，SSIG可以置1

- 可以忽略SS引脚。如果SSIG为0，在连续传输之间SS引脚保持低电平有效。这种方式适合具有单个固定主设备和单个固定从设备之间驱动MISO数据线的系统。

SPI模块配置及时序

--通过SS修改模式

如果SPEN为1，SSIG为0且MSTR为1，则SPI为主模式。

- 通过P2M1和P2M0寄存器（用于设备P2端口的输入/输出模式），将SS引脚配置为输入或者准双向模式。

- 在该模式下，另外一个主设备可以将该引脚驱动为低，从而将该器件选择为SPI从设备，并向该从设备发送数据。

- 为了避免总线冲突，SPI执行下面的行为：

- 清零MSTR，并且变为从设备。同时，将MOSI和SCLK强制作为输入模式，而MISO作为输出模式。

- 将SPSTAT寄存器的SPIF标志置为1。如果已经使能SPI中断，则产生SPI中断。

SPI模块配置及时序

--写冲突

SPI在发送数据时，为单级缓冲方式；而在接收数据时，为双缓冲方式。

- **如果在发送数据的过程中，向移位寄存器写数据，则将WCOL位置1表示数据冲突。**
 - **在这种情况下，继续发送完当前的传输的数据，但是新写入的数据丢失。**

SPI模块配置及时序

--写冲突

- 由于主设备拥有数据传输的控制能力,因此主设备发生写冲突的情况是比较少见的。
 - 但是,从设备有可能发生写冲突。这是因为当主设备启动数据传输时,从设备无法控制数据的传输过程。
- 当从设备接收到数据时,将接收到的数据发送到一个并行读数据缓冲区。
 - 这样,就释放了移位寄存器用于接收下一个数据。
 - 必须在下一个字符完全移入之前从数据寄存器中读出接收到的数据。否则,将丢失前一个数据。

SPI模块配置及时序

--数据模式时序

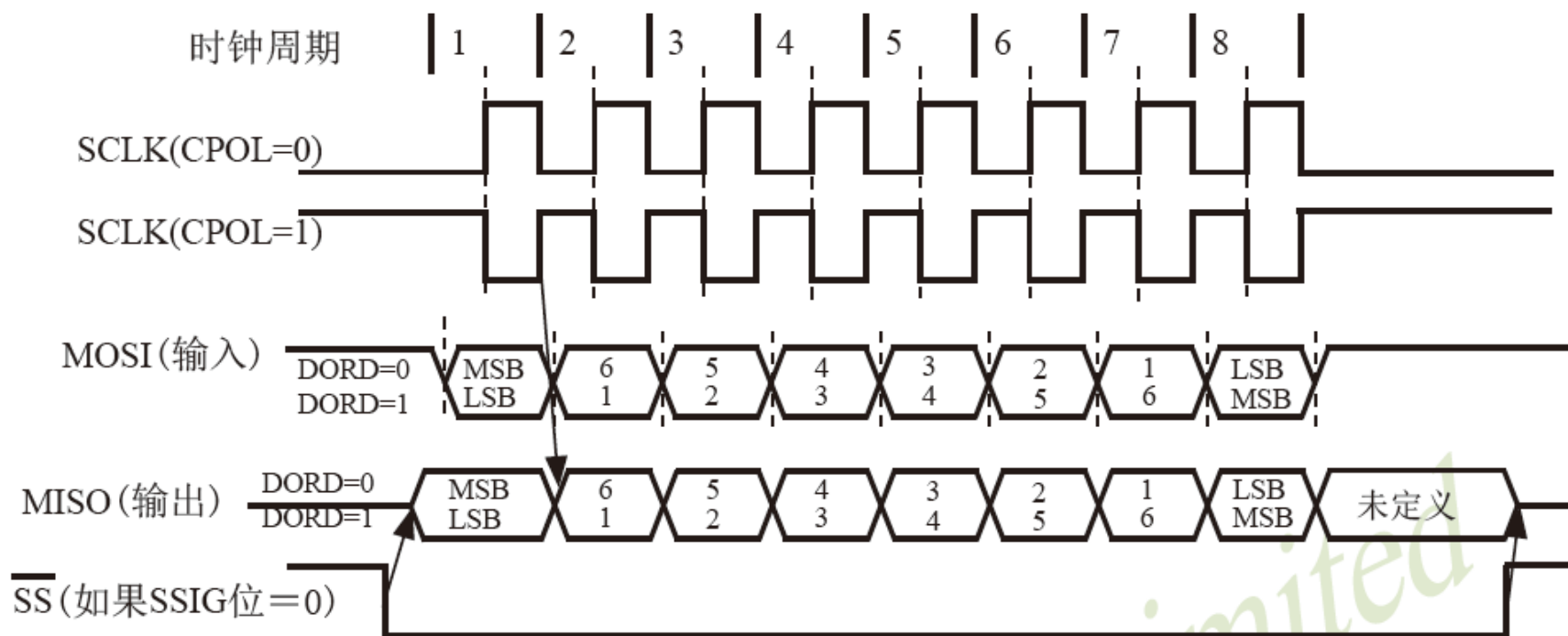
通过时钟相位控制位CPHA，允许用户设置采样和改变数据的时钟边沿。此外，时钟极性比特控制位CPOL，允许用户设置时钟的极性。

■ 从后面图中可以看出：

- 当CPOL为0时，在空闲状态下，SCLK为低电平；
- 当CPOL为1时，在空闲状态下，SCLK为高电平。

SPI模块配置及时序

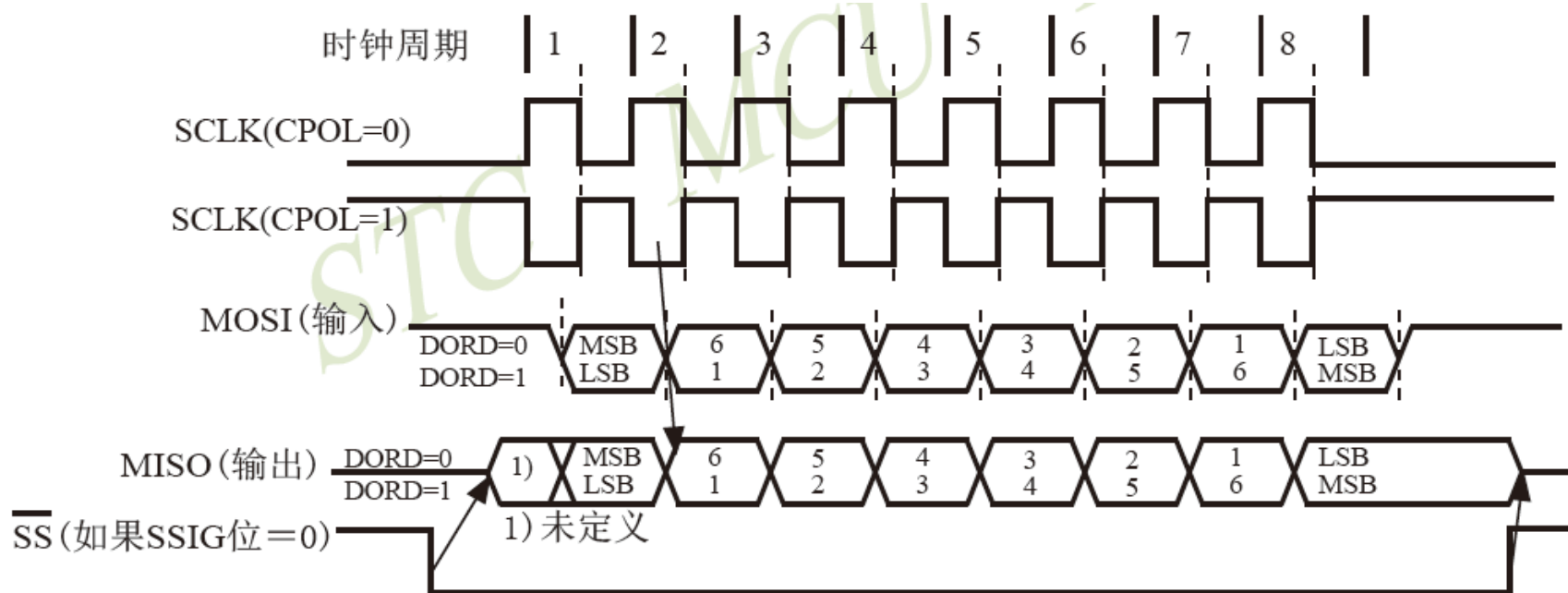
--数据模式时序



CPHA为0时从模式数据传输时序图

SPI模块配置及时序

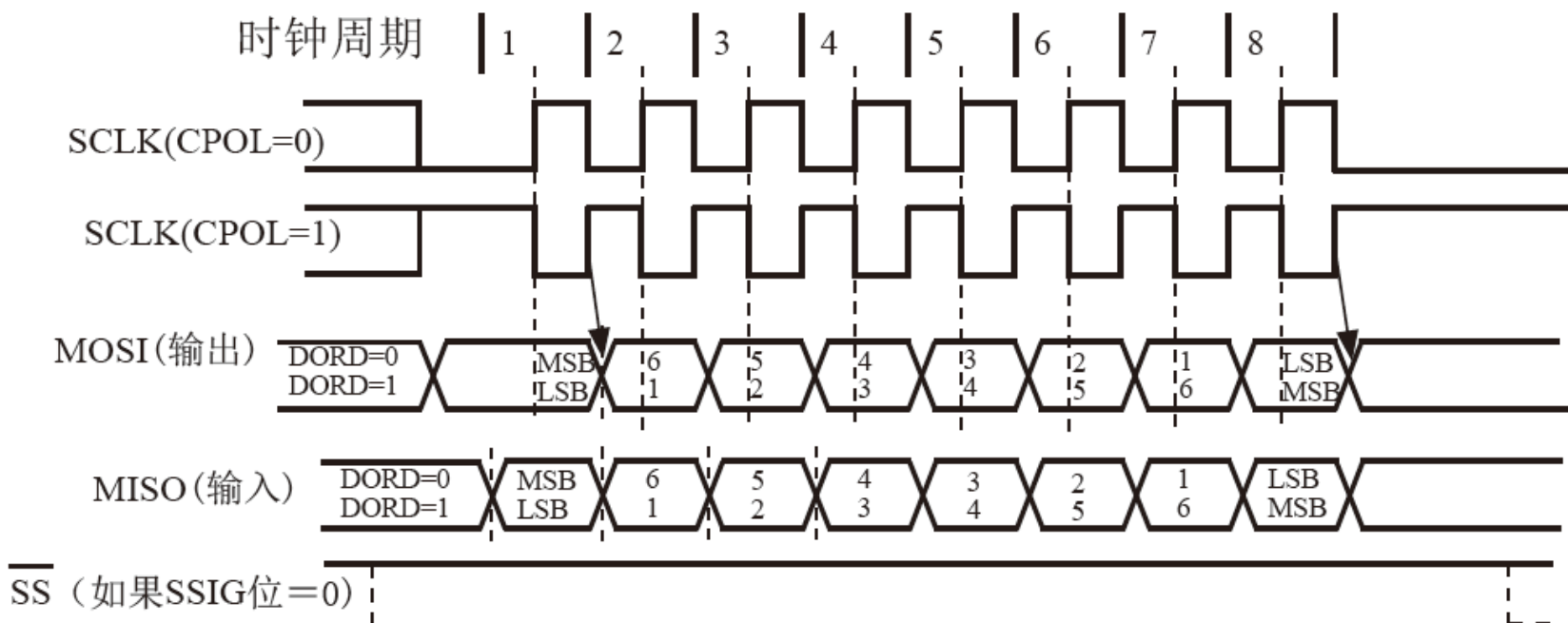
--数据模式时序



CPHA为1时从模式数据传输时序图

SPI模块配置及时序

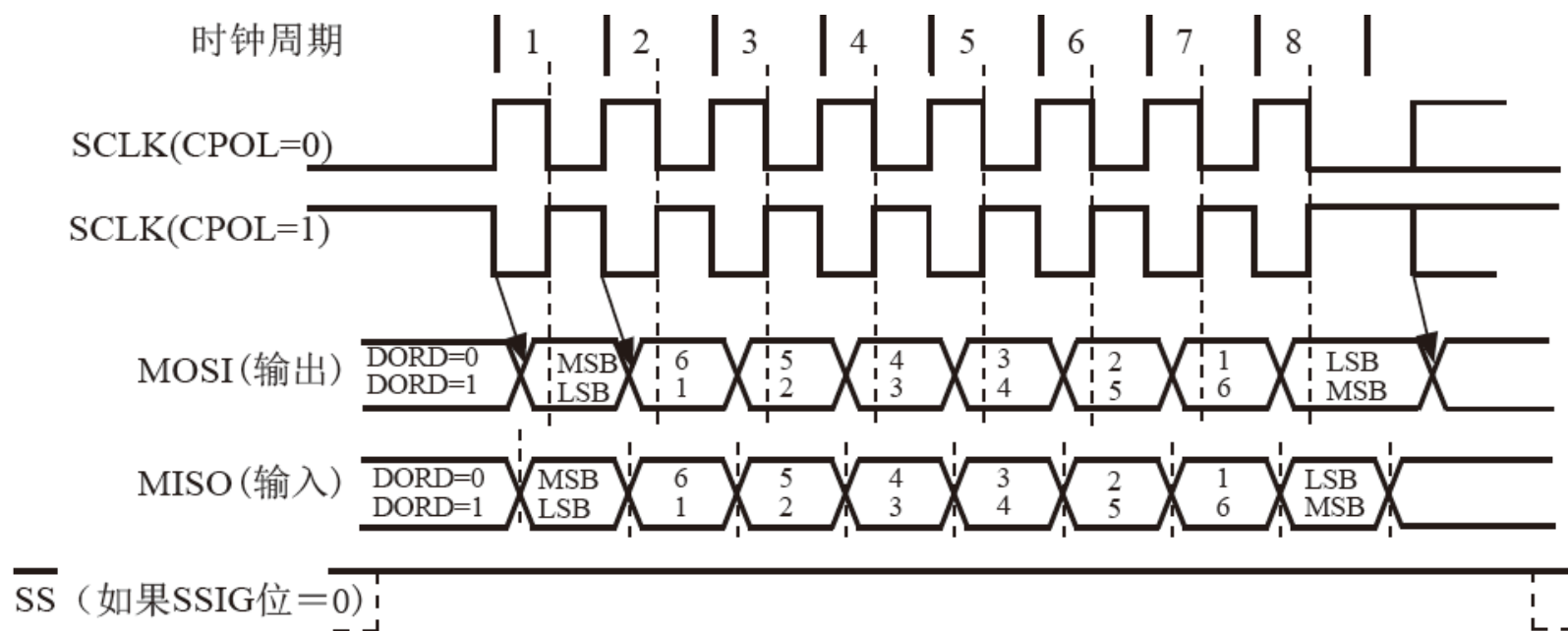
--数据模式时序



CPHA为0时主模式数据传输时序图

SPI模块配置及时序

--数据模式时序



CPHA为1时主模式数据传输时序图

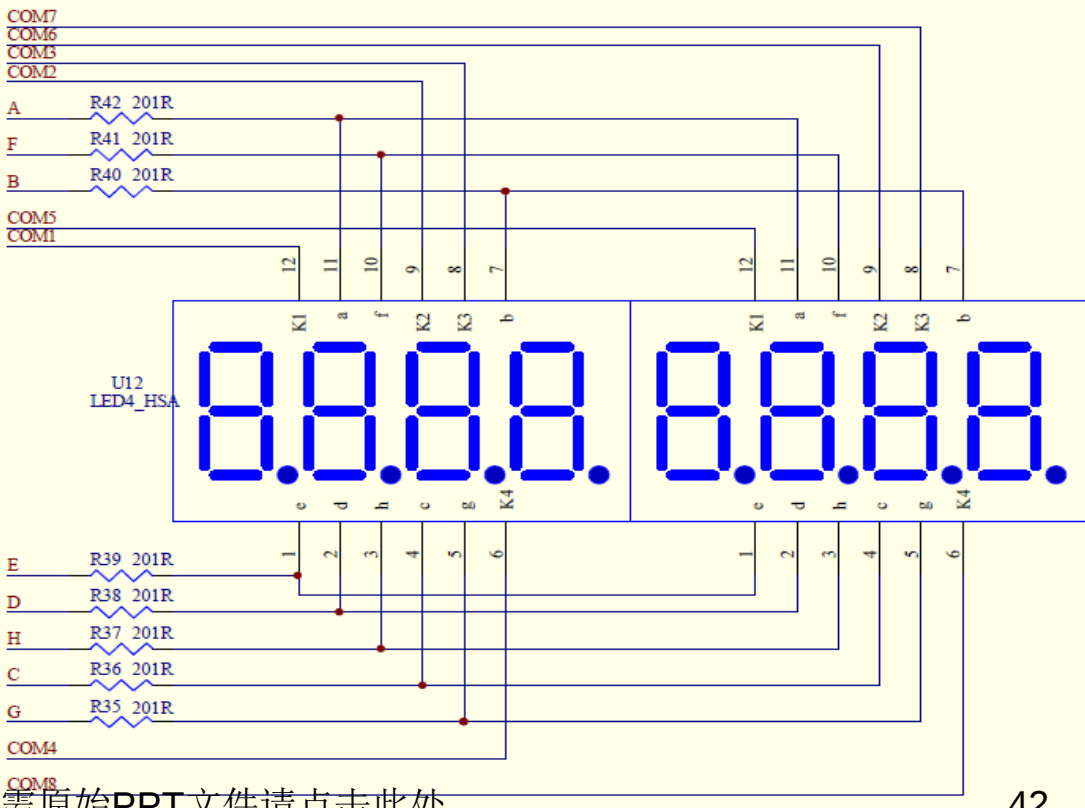
SPI模块设计实例

--系统控制电路原理

在STC学习板上，为了减少控制7段数码管所使用的引脚的数目，使用两片74HC595对7段数码管进行控制。

- 其中一片74HC595用于产生控制8个7段数码管的管选信号COM1~COM8；
- 另一片74HC595用于为每个数码管产生段控制信号A~H，其中一个信号用于控制显示小数点。
 - 与七段数码管连接的信号线，分别通过8个电阻进行限流。
- 74HC595器件提供了SPI接口，与单片机上的P4.3/SCLK、P5.4/SS和P4.0/MOSI引脚连接在一起。

- 

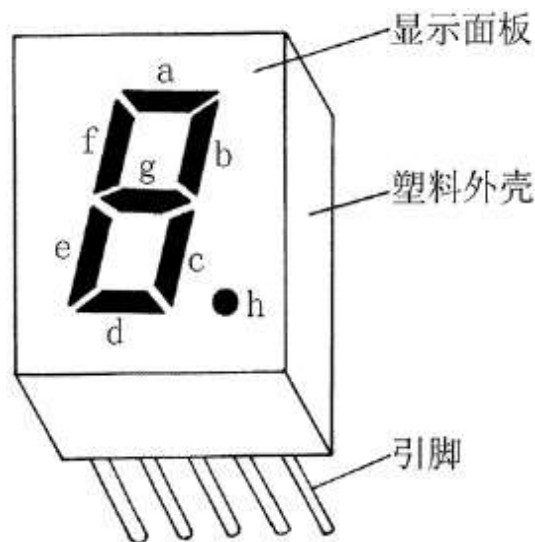
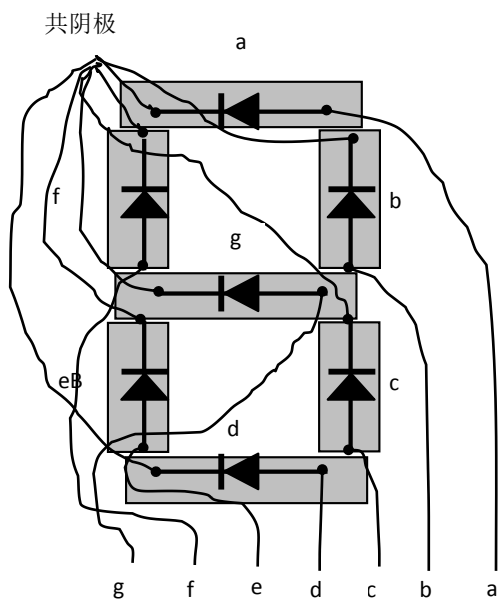


SPI模块设计实例

--7段数码管原理

单个共阴极七段数码管控制原理

- 7段数码管亮灭控制的最基本原理就是当有电流流过7段数码管a、b、c、d、e、f、g的某一段时，该段就发光。



SPI模块设计实例

--7段数码管原理

- $V_a - V_{\text{公共端}} < V_{\text{TH}}$ 时，a段灭；否则，a段亮。
- $V_b - V_{\text{公共端}} < V_{\text{TH}}$ 时，b段灭；否则，b段亮。
- $V_c - V_{\text{公共端}} < V_{\text{TH}}$ 时，c段灭；否则，c段亮。
- $V_d - V_{\text{公共端}} < V_{\text{TH}}$ 时，d段灭；否则，d段亮。
- $V_e - V_{\text{公共端}} < V_{\text{TH}}$ 时，e段灭；否则，e段亮。
- $V_f - V_{\text{公共端}} < V_{\text{TH}}$ 时，f段灭；否则，f段亮。
- $V_g - V_{\text{公共端}} < V_{\text{TH}}$ 时，g段灭；否则，g段亮。

注：V_{TH}为七段数码管各段的门限电压。

SPI模块设计实例

--7段数码管原理

控制7段数码管显示不同的数字和字母时，只要给不同段施加高电平（逻辑“1”）即可。将二进制码编码转换为所对应的7段码

x_3	x_2	x_1	x_0	g	f	e	d	c	b	a
0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1	0	1	1
0	0	1	1	1	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	1
0	1	1	1	0	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1	0	0
1	1	0	0	0	1	1	1	0	0	1
1	1	0	1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1	0	0	1
1	1	1	1	1	1	1	0	0	0	1
1	1	1	1	1	1	1	0	0	0	1

SPI模块设计实例

--多个共阴极7段数码管控制原理

在7段数码管上显示不同的数字

/字母时，满足条件： $COM_i=0$

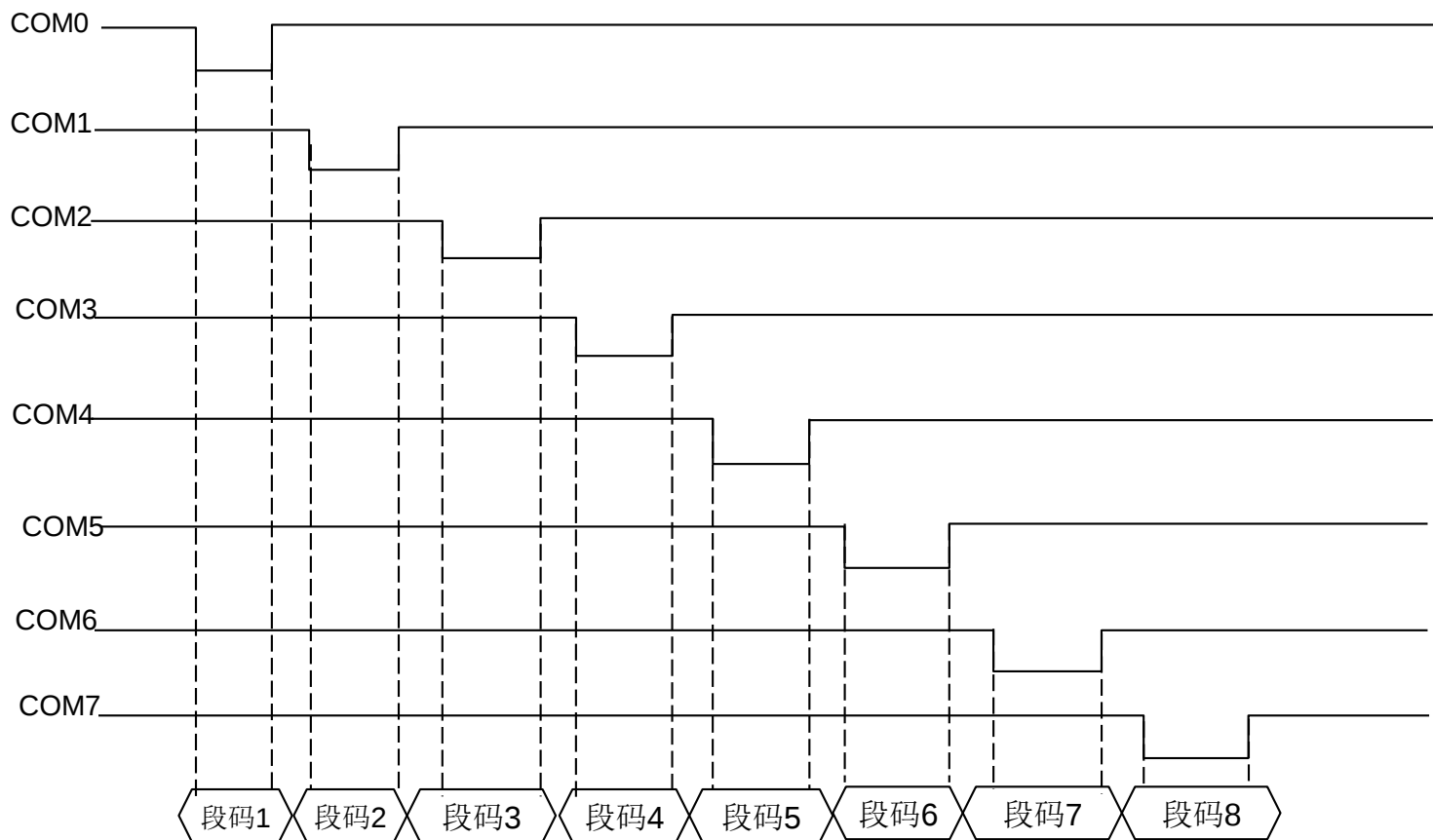
($i=0,1,2,3,4,5,6,7$) 时，对应给出合适的A~H信号。也就是，在不同的时刻，使得 $i=0\sim7$ 快速的进行递增变化，如图所示。

导通频率大约在100KHz的量级。

导通频率越高，人眼看到的数字/字母越稳定。

SPI模块设计实例

--多个共阴极7段数码管控制原理



SPI模块设计实例

--74HCT595原理



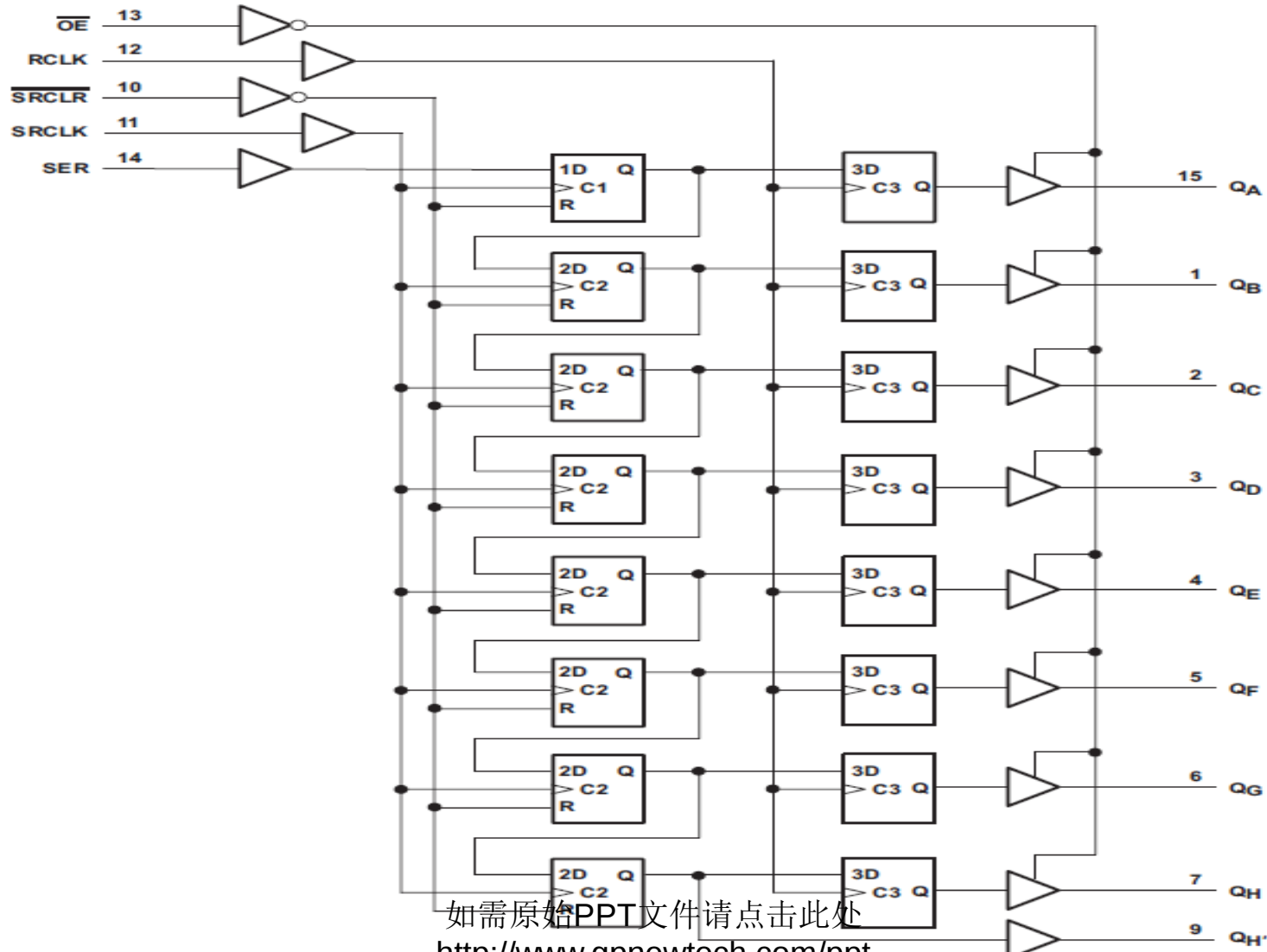
74HCT595芯片是一个带有3态输出寄存器的8位移位寄存器

74HCT595功能表

输入					输出
SER	SRCLK	SRCLR	RCLK	OE	
X	X	X	X	H	禁止 $Q_A \sim Q_H$ 输出
X	X	X	X	L	使能 $Q_A \sim Q_H$ 输出
X	X	L	X	X	清除移位寄存器
L	↑	H	X	X	第一个移位寄存器变低，其他不变
H	↑	H	X	X	第一个移位寄存器变高，其他不变
X	X	X	↑	↑	移位寄存器的数据保存在存储寄存器中

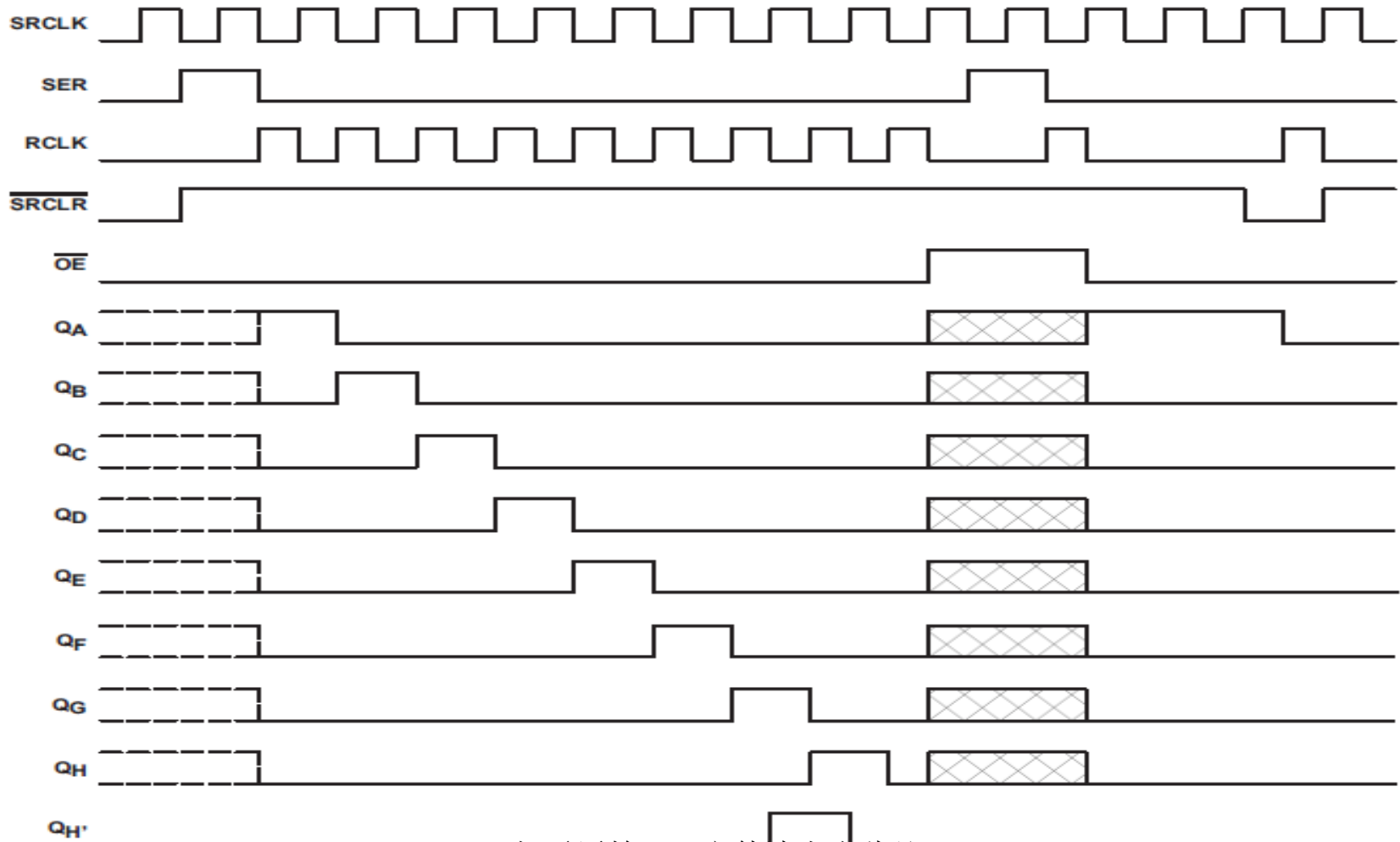
SPI模块设计实例

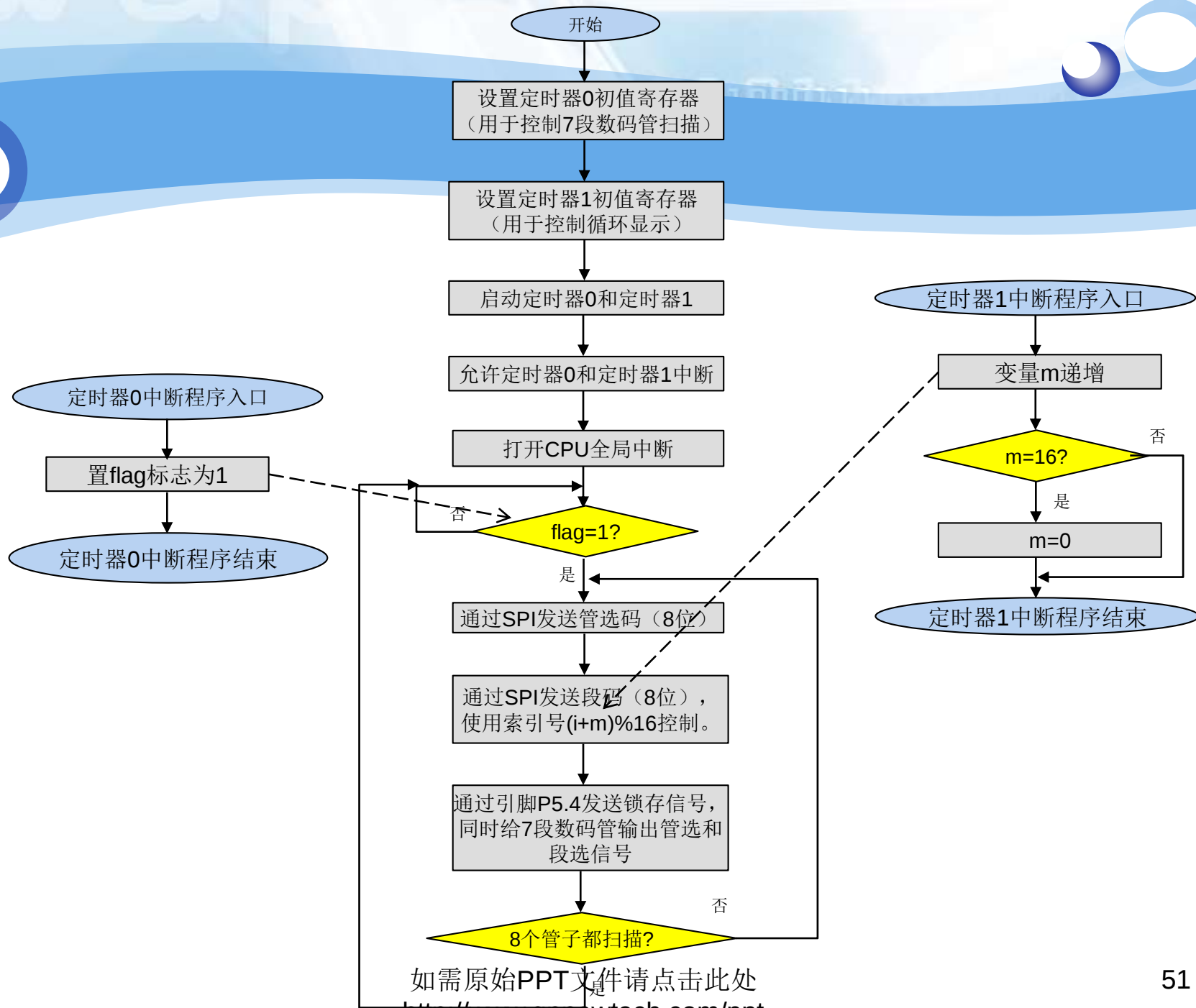
--74HCT595原理



SPI模块设计实例

--74HCT595原理





SPI模块设计实例

--程序具体实现



【例】 通过SPI接口和74HCT595芯片控制7段数码管C语言描述的例子。

```
#define TIMS 65500           //定义定时器0的计数初值
#define TIMS1 3036          //定义定时器1的计数初值
#define SSIG    1           //定义SPCTL寄存器SSIG位的值
#define SPEN    1           //定义SPCTL寄存器SPEN位的值，使能SPI
#define DORD    0           //定义SPCTL寄存器DORD位的值，先送MSB
#define MSTR    1           //定义SPCTL寄存器MSTR位的值，SPI为主机
#define CPOL    1           //定义SPCTL寄存器CPOL位的值，空闲为高电平
#define CPHA    1           //定义SPCTL寄存器CPHA位的值，前沿驱动数据
```

SPI模块设计实例

--程序具体实现

```
#define SPR1    0           //与SPECTL寄存器SPR0一起确定SPI的时钟频率
#define SPR0    0           //SPI时钟频率为CPU时钟的1/4
#define SPEED_4  0
#define SPEED_16 1
#define SPEED_64 2
#define SPEED_128 3
#define SPIF     0x80       //定义SPSTAT寄存器SPIF标志的值
#define WCOL     0x40       //定义SPSTAT寄存器WCOL标志的值
sfr SPSTAT = 0xCD;         //定义SPSTAT寄存器的地址0xCD
sfr SPCTL  = 0xCE;         //定义SPCTL寄存器的地址0xCE
sfr SPDAT  = 0xCF;         //定义SPDAT寄存器的地址0xCF
```

SPI模块设计实例

--程序具体实现

```
sfr AUXR    =0x8E;           //定义AUXR寄存器的地址0x8E
sfr AUXR1   =0xA2;           //定义AUXR1寄存器的地址0xA2
sfr CLK_DIV =0x97;           //定义CLK_DIV寄存器的地址0x97
sfr P5      =0xC8;           //定义P5端口寄存器的地址0xC8
sbit HC595_RCLK=P5^4;        //定义P5.4引脚
```

SPI模块设计实例

--程序具体实现

main.c文件

```
#include "reg51.h"
```

```
#include "spi.h"
```

//包含自定义头文件

//t_display数组保存着0~9 , A~F的段码 , 顺序h,g,f,e,d,c,b,a

```
unsigned char code t_display[16]={0x3F,0x06,0x5B,0x4F,  
                                0x66,0x6D,0x7D,0x07,  
                                0x7F,0x6F,0x77,0x7C,  
                                0x39,0x5E,0x79,0x71};
```

//T-COM数组保存着管选码的反码 , 在一个时刻只有一个管选信号为低

```
unsigned char code T_COM[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
```

```
bit flag=0;
```

//定义全局位变量flag

SPI模块设计实例

--程序具体实现

```
unsigned m=0; //定义全局无符号变量m
void SPI_SendByte(unsigned char dat) //定义SPI数据发送函数
{
    SPSTAT=SPIF+WCOL; //写“1” 清零SPSTAT寄存器内容
    SPDAT=dat; //dat写入SPDATSPI数据寄存器
    while((SPSTAT & SPIF)==0); //判断发送是否完成，没有则等待
    SPSTAT=SPIF+WCOL; //写“1” 清零SPSTAT寄存器内容
}
//定义7段数码管的函数seg7scan，index1参数控制管选，Index2控制段码
void seg7scan(unsigned char index1,unsigned char index2)
{ SPI_SendByte(~T_COM[index1]); //向74HCT595（U5）写入管选信号
```

SPI模块设计实例

--程序具体实现



```
SPI_SendByte(t_display[index2]); //向74HCT595 ( U6 ) 写入段码数据
```

```
HC595_RCLK=1;
```

```
//通过P5.4端口向两片595发数据锁存
```

```
HC595_RCLK=0;
```

```
//上升沿有效
```

```
}
```

```
void timer_0() interrupt 1
```

```
//声明定时器0的中断服务程序
```

```
{
```

```
flag=1;
```

```
//置flag标志为1
```

```
}
```

```
void timer_1() interrupt 3
```

```
//声明定时器1的中断服务程序
```

```
{ P46=!P46;
```

```
//P4.6引脚取反
```

```
m++;
```

```
//全局变量m递增
```

```
if(m==16) m=0;
```

```
//如果m等于16，则m置为0
```

```
}
```

SPI模块设计实例

--程序具体实现

```
void main()
```

```
{
```

```
    unsigned char i=0;           //定义本地字符型变量char
```

```
    SPCTL=(SSIG<<7)+(SPEN<<6)+(DORD<<5)+(MSTR<<4)  
          +(CPOL<<3)+(CPHA<<2)+SPEED_4;
```

```
    //给寄存器SPCTL赋值
```

```
    CLK_DIV=0x03;                //主时钟8分频作为SYSclk频率
```

```
    TL0=TIMS;                    //TIMS写入定时器0低8位寄存器TL0
```

```
    TH0=TIMS>>8;                //TIMS写入定时器0高8位寄存器TH0
```

```
    TL1=TIMS1;                  //TIMS1写入定时器1低8位寄存器TL1
```

```
    TH1=TIMS1>>8;              //TIMS1写入定时器1高8位寄存器TH1
```

```
    AUXR&=0x3F;                //定时器0和1是12分频
```

SPI模块设计实例

--程序具体实现

```
AUXR1=0x08;           //将SPI接口信号线切换到第3组引脚上
TMOD=0x00;             //定时器0/1 , 16位重加载定时器模式
TR0=1;                 //启动定时器0
TR1=1;                 //启动定时器1
ET0=1;                 //允许定时器0溢出中断
ET1=1;                 //允许定时器1溢出中断
EA=1;                  //CPU允许响应中断请求
while(1)               //无限循环
{
    if(flag==1)         //如果flag为1 , 表示定时器0中断
```

SPI模块设计实例

--程序具体实现

```
{  
    flag=0;                                //将flag标志清零  
    for(i=0;i<8;i++)                        //轮流导通7段数码管，需要8次  
    {  
        seg7scan(i,(m+i)%16); //控制其中一个数码管，送管选和段码  
    }                                       //（ m+i ） %16为了控制每个7段数码管  
}                                           //上显示的数字  
}  
}
```