第15章 STC单片机SPI原理及实现

何宾

2018.03

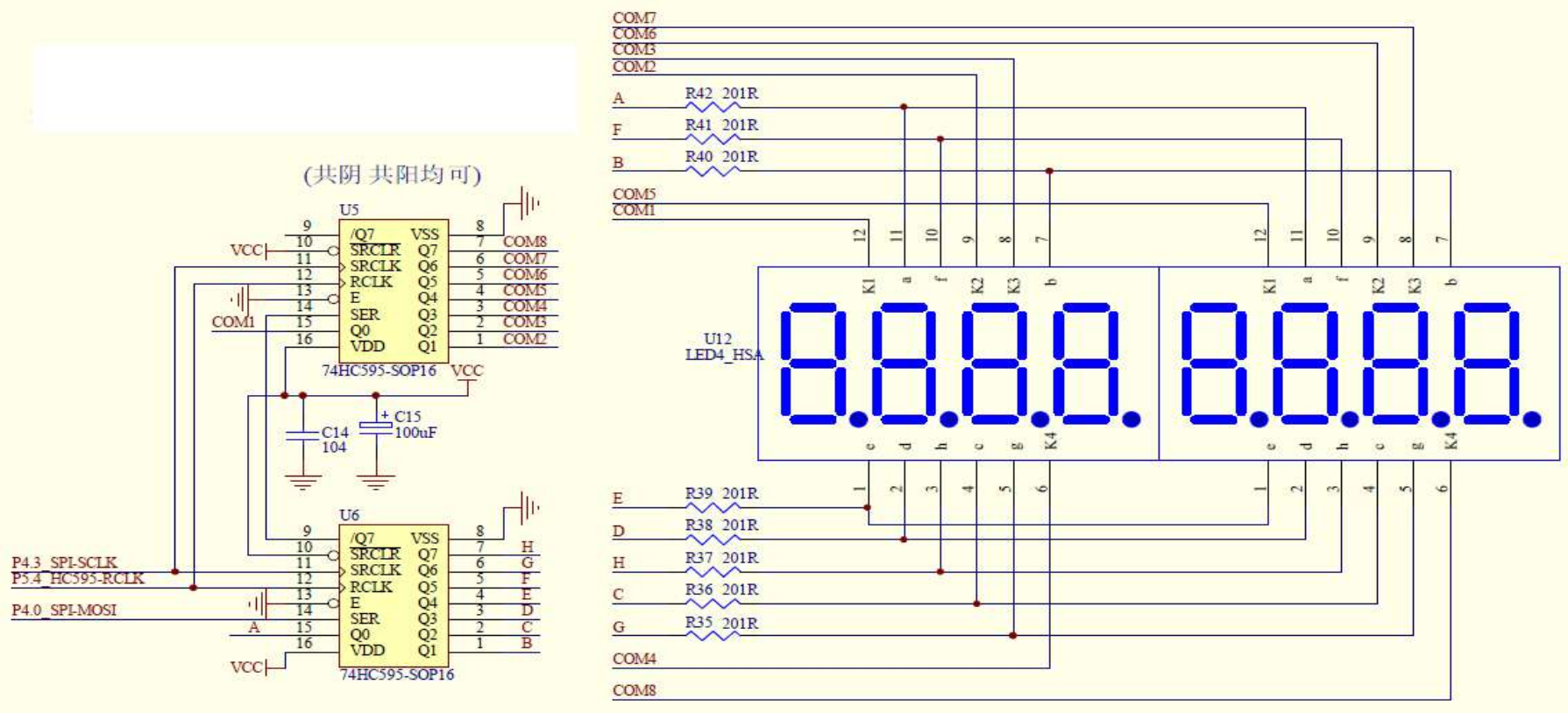
SPI模块设计实例
--I/O扩展和七段数码管

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
--系统控制电路原理

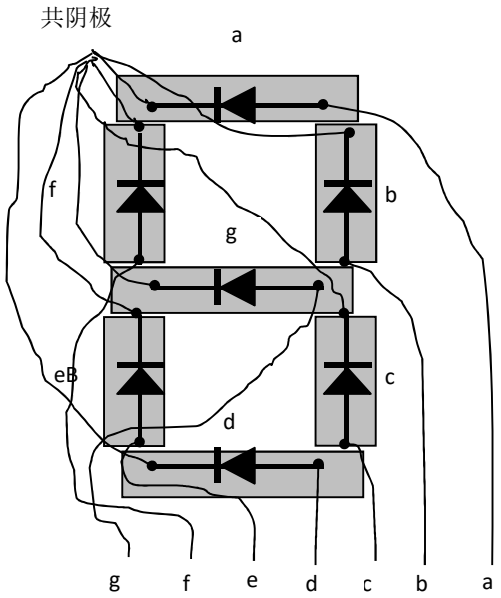
- 从下图可以看出，实现设计目标的关键是掌握7段数码管和74HC595的工作原理。



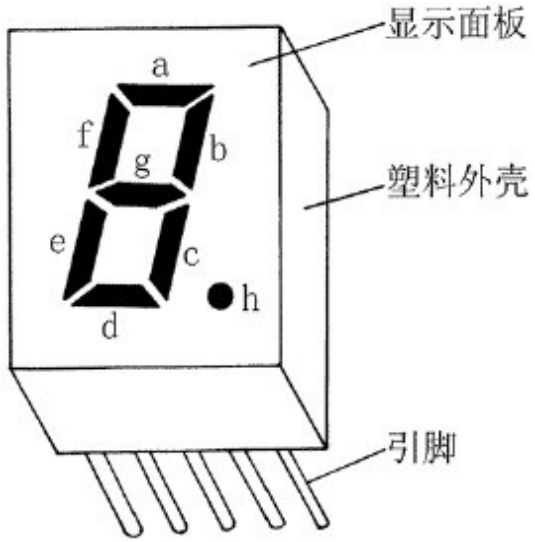
SPI模块设计实例
--7段数码管原理

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。



(a) 共阴极7段数码管原理



（b）7段数码管外观

SPI模块设计实例
--7段数码管原理

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

注：VTH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**

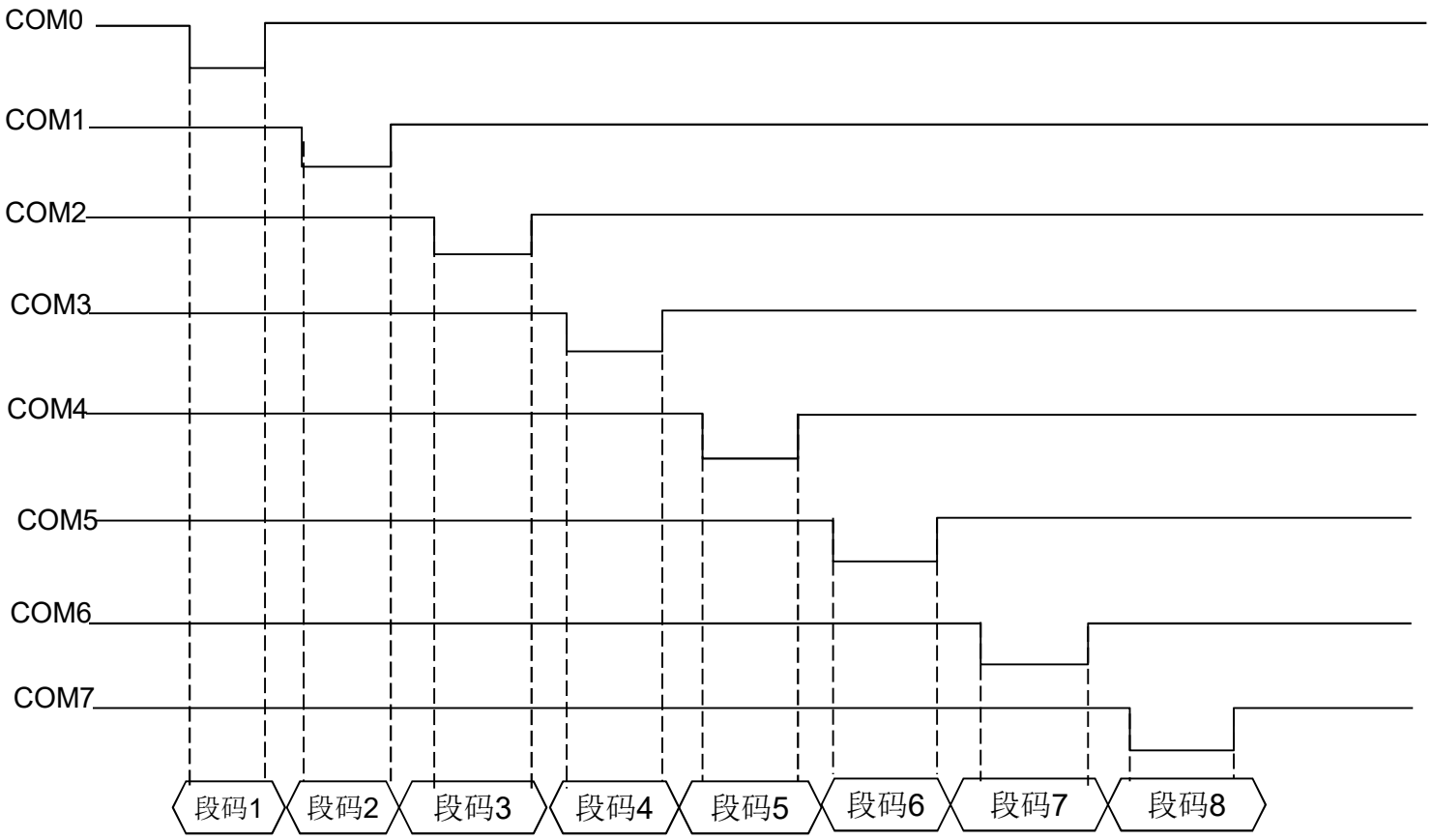
SPI模块设计实例
--多个共阴极7段数码管控制原理

在7段数码管上显示不同的数字/字母时，满足条件：COMi=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=0~7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



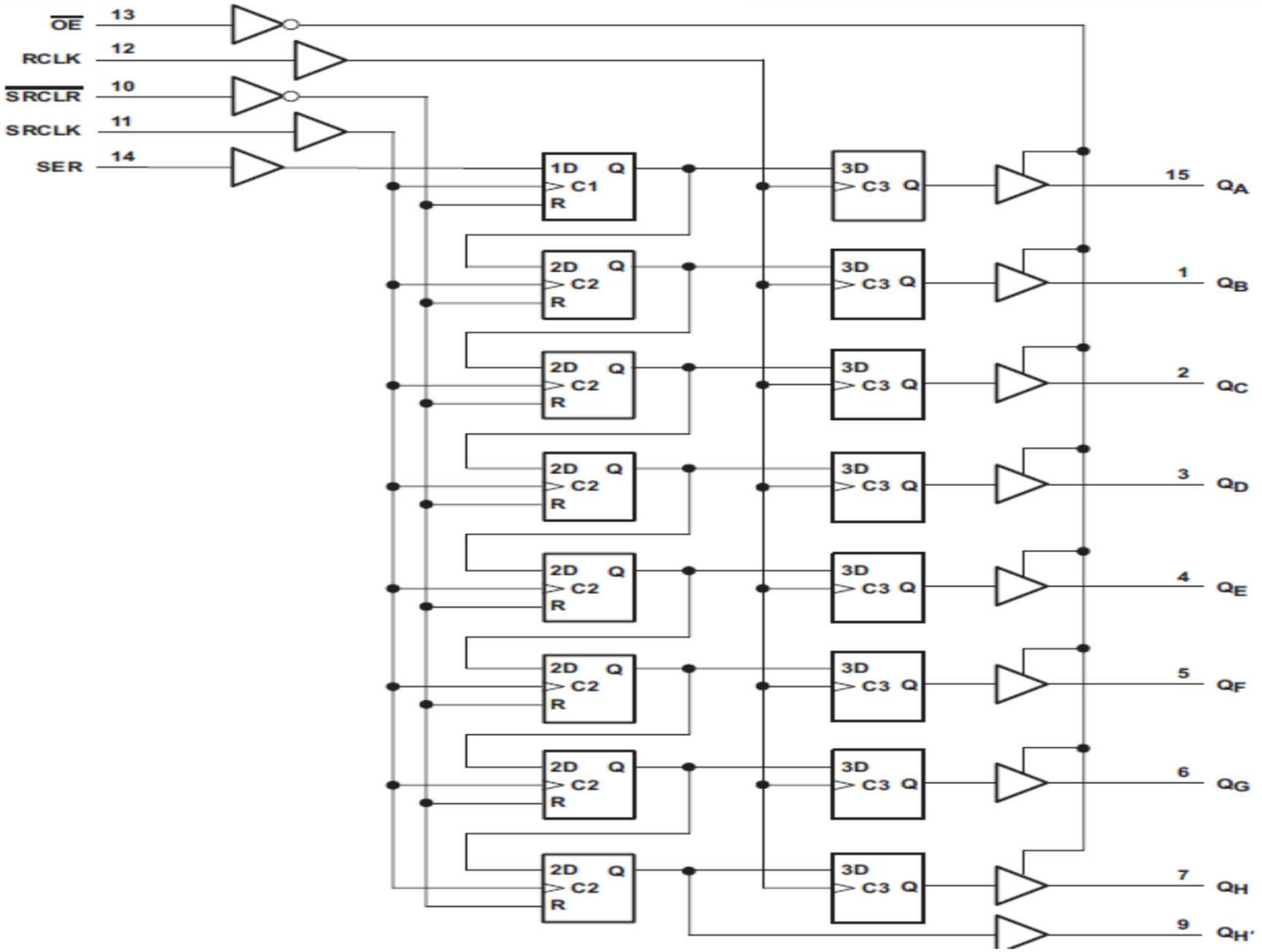
SPI模块设计实例
--74HCT595原理

74HCT595芯片是一个带有3态输出寄存器的8位移位寄存器

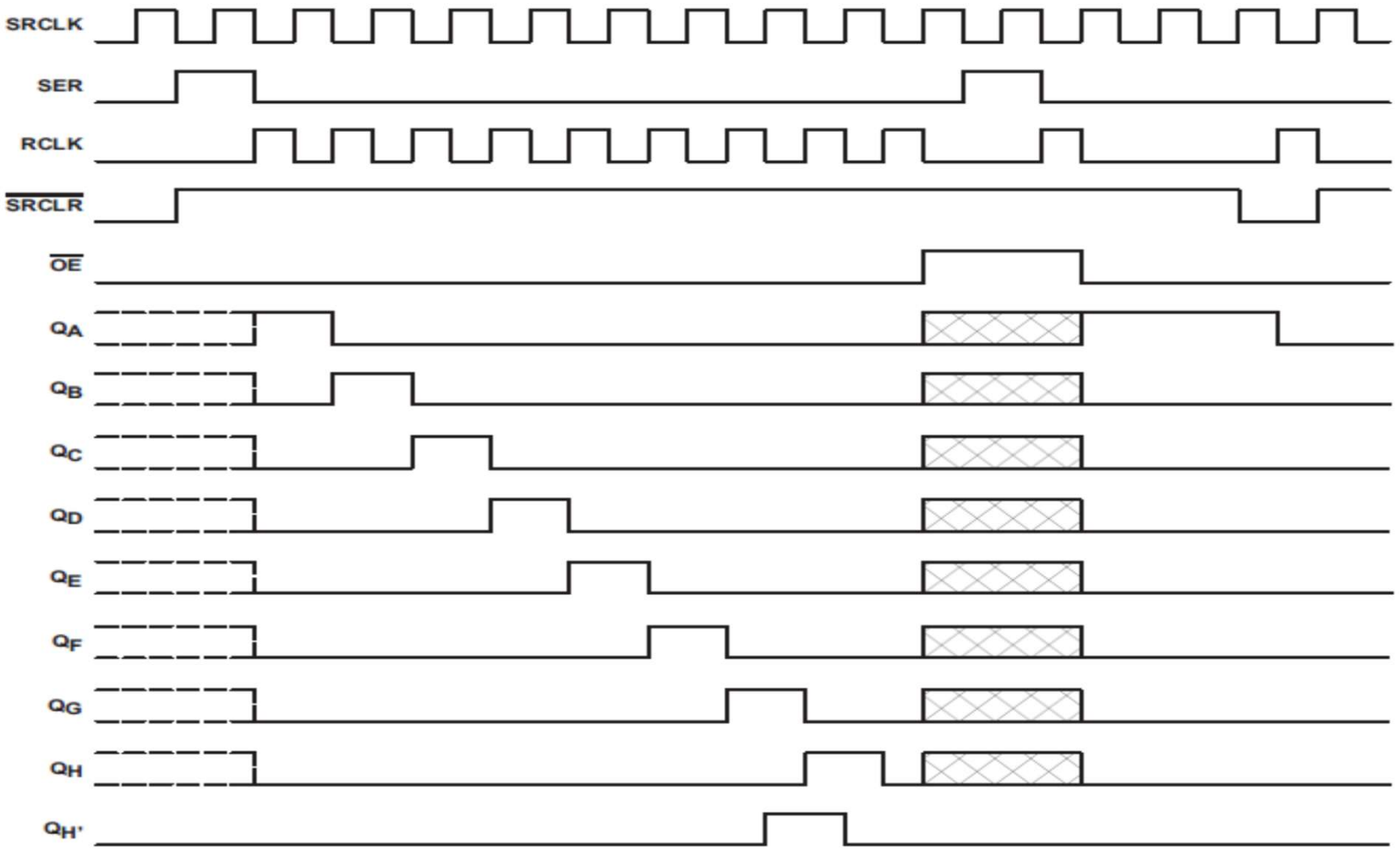
74HCT595功能表

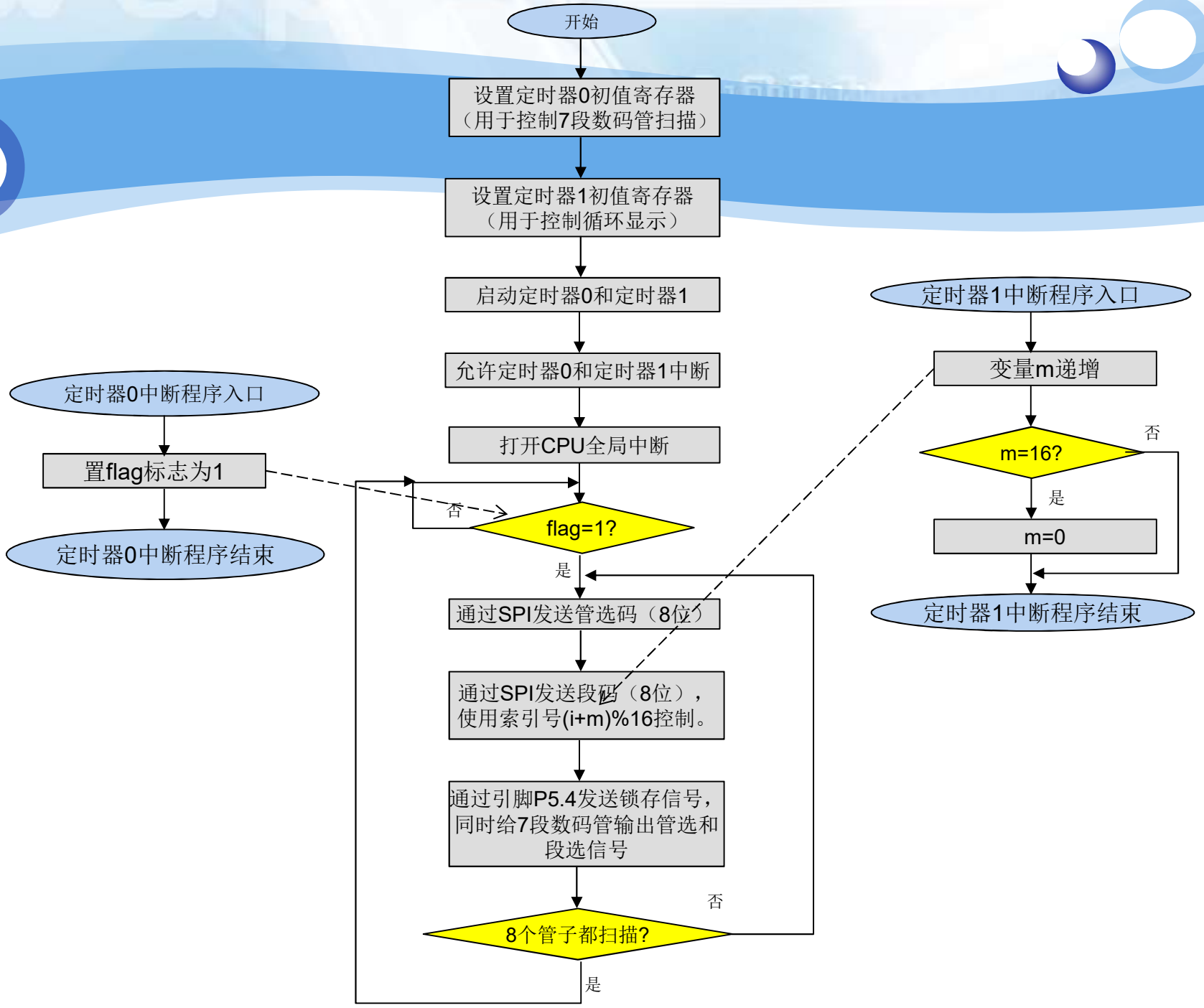
输入					输出
SER	SRCLK	SRCLR	RCLK	OE	
X	X	X	X	H	禁止Q_A~Q_H输出
X	X	X	X	L	使能Q_A~Q_H输出
X	X	L	X	X	清除移位寄存器
L	↑	H	X	X	第一个移位寄存器变低，其他不变
H	↑	H	X	X	第一个移位寄存器变高，其他不变
X	X	X	↑	↑	移位寄存器的数据保存在存储寄存器中

SPI模块设计实例
--74HCT595原理



SPI模块设计实例
--74HCT595原理




开始
设置定时器0初值寄存器
（用于控制7段数码管扫描）
设置定时器1初值寄存器
（用于控制循环显示）
启动定时器0和定时器1
允许定时器0和定时器1中断
打开CPU全局中断
flag=1?
否
是
通过SPI发送管选码（8位）
通过SPI发送段码（8位），
使用索引号(i+m)%16控制。
通过引脚P5.4发送锁存信号，
同时给7段数码管输出管选和
段选信号
8个管子都扫描?
否
是
定时器0中断程序入口
置flag标志为1
定时器0中断程序结束
定时器1中断程序入口
变量m递增
m=16?
否
是
m=0
定时器1中断程序结束

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
    }
}
```