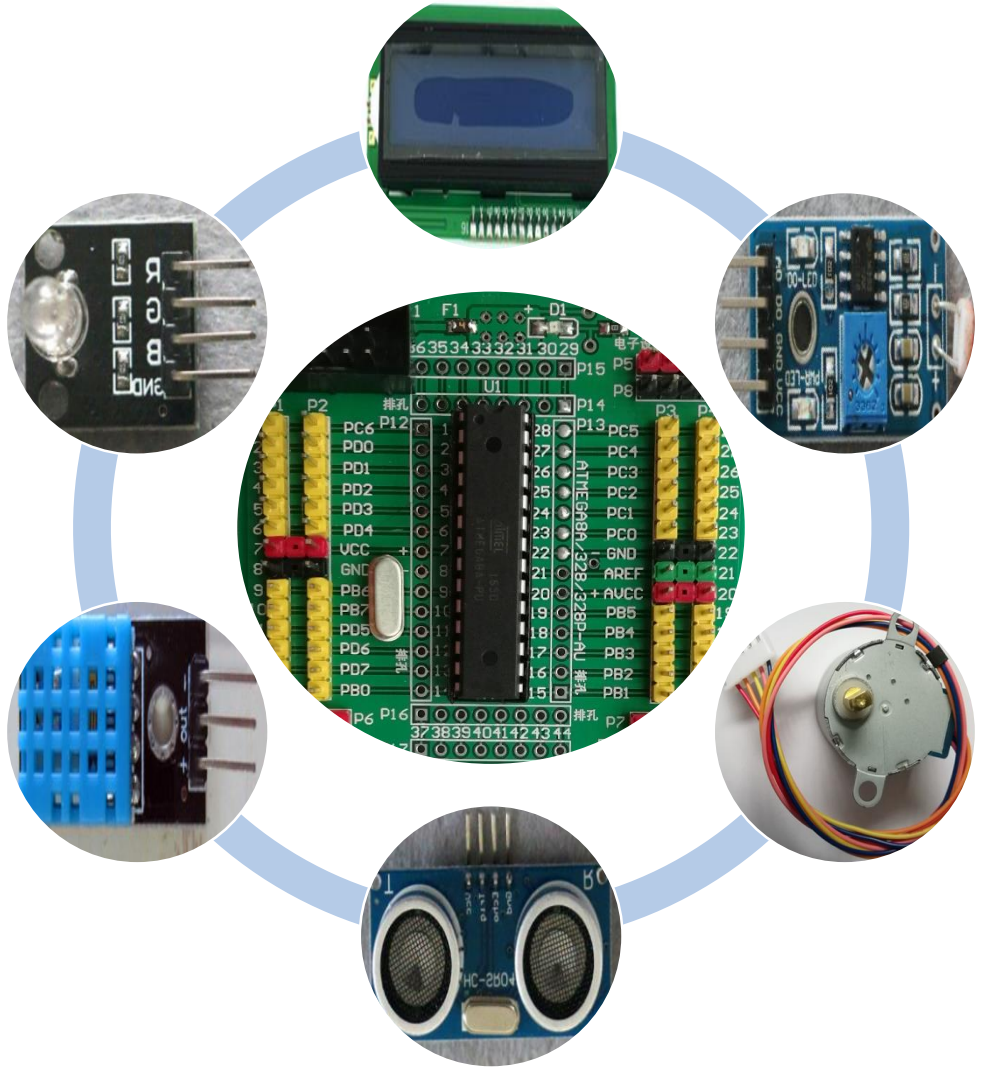
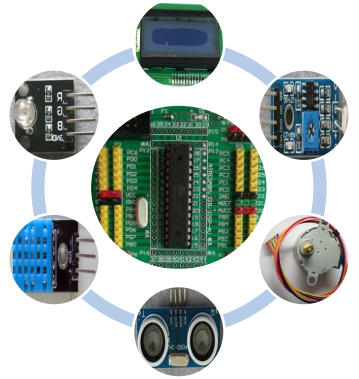
电子设计实践基础

MCU中断、数码管、按键阵列及其编程



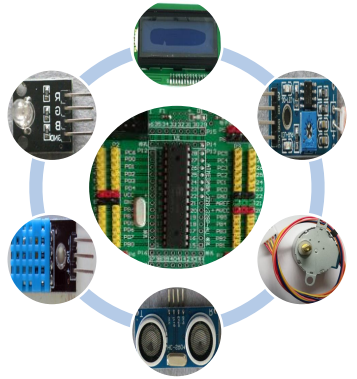
上节课内容回顾

- MCU芯片(ATmega8A)与其程序编写
  - #include <avr/io.h>
  - #include <util/delay.h>　//需前置#define F_CPU…
  - IO端口控制与编程总结：DDRx，PORTx和PINx
  - C语言的位运算：(PINB & (1<<PINB1))>>1
  - RGB LED及其编程：流水灯、呼吸灯
  - 一位触摸开关及其编程：控制流水/呼吸灯



本节课主要内容

- MCU中断、七段数码管与按键阵列
  - ATmega8A的中断和编程
  - 七段数码管及其编程
  - 按键阵列(小键盘)及其编程



统计触摸开关的开关次数

1：高电平

0：低电平

R2=0,R1=1

R2=1,R1=0

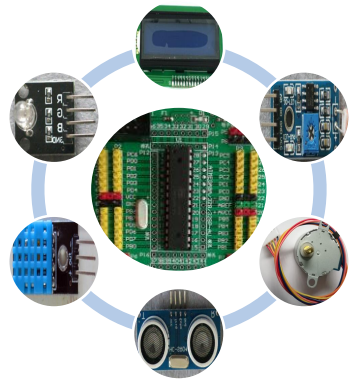
2

1

R1

R2

工作时钟



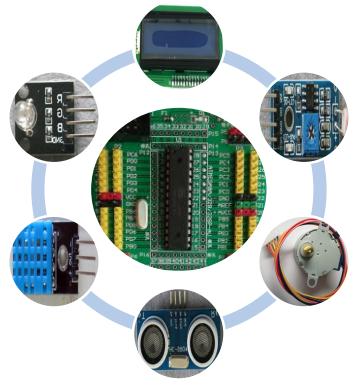
统计触摸开关次数的关键程序

```
while (1)
{
    tpr2 = tpr1;
    tpr1 = (PINB & (1<<PINB1))>>1;
    if(tpr2==0 && tpr1==1)counter++;
    switch(counter)
    {
        case 1 : PORTC = (1<<PORTC2);break;
        case 2 : PORTC = (1<<PORTC1);break;
        case 3 : PORTC = (1<<PORTC0);break;
        default :PORTC =0;counter = 0;
    }
}
```

丢失触摸开关的动作?

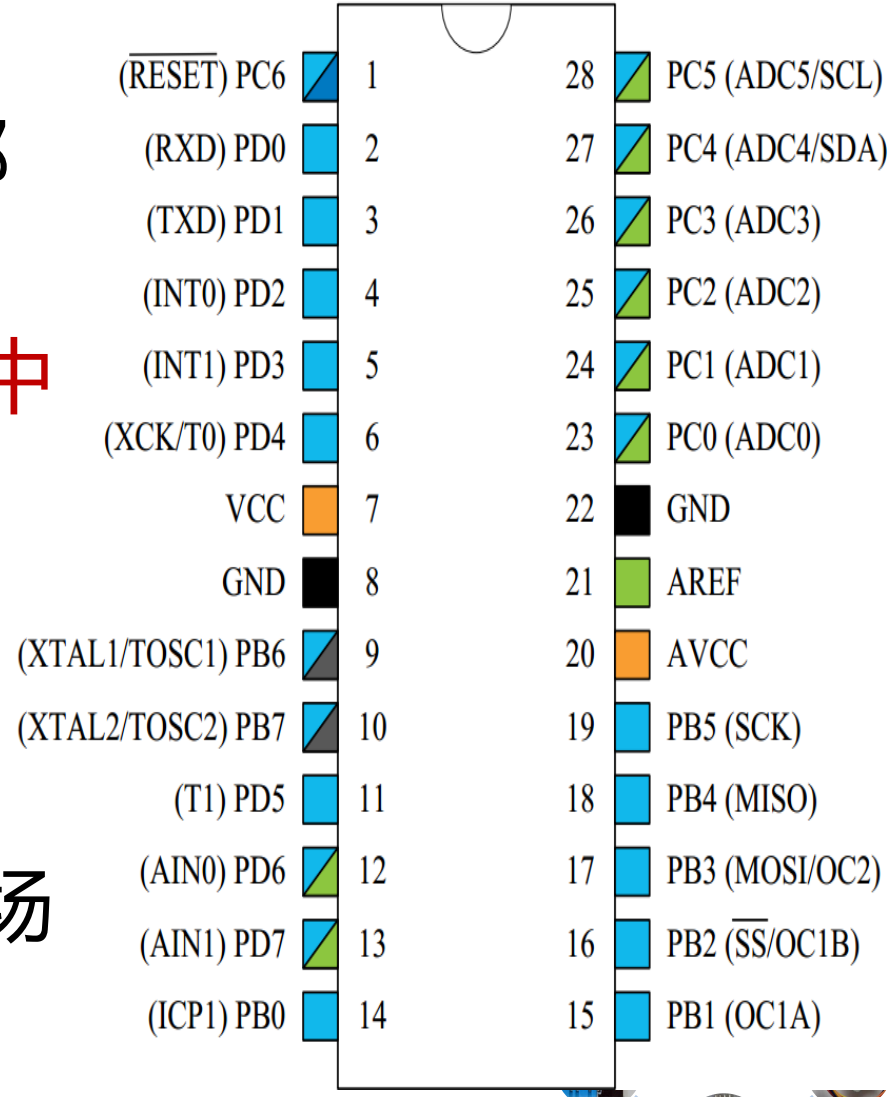
中断

如执行时间太长



ATmega8A的中断概念

- **允许中断**/开中断/中断使能
  - 开启全局、特定中断源：内部/外部
- **中断向量表**
  - 中断源入口地址，其作用是跳转到中断服务程序起始的地址
- 编写**中断服务程序**(类似函数)
  - 中断后处理特定任务的一段代码
- 中断现场的**保护与恢复**
  - 保存断点数据，任务结束后恢复现场
- 禁止中断-**关中断**



ATmega8A中断的过程

开中断

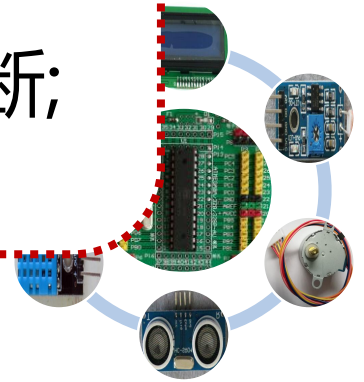
```
while(1)
{
  常规任务，…
}
```

出现中断

中断向量表：
CPU跳转到表中对应中断的入口地址处执行（跳转语句）

中断服务过程{
关中断，保护现场；
中断处理；
恢复现场，开中断；
}

中断处理结束



ATmega8A全局中断的开或关

■**SREG**-Status REGister：IO空间地址：0x3F

Bit	7	6	5	4	3	2	1	0
SREG	I	T	H	S	V	N	Z	C
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

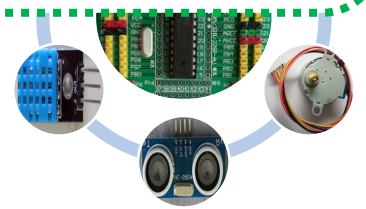
0:关中断

1:开中断

SREG &= 0x7F

SREG |= 0x80

```
#include <avr/interrupt.h>
sei(); //开中断:set interrupt
cli(); //关中断:clear interrupt
```



ATmega8A的中断向量表

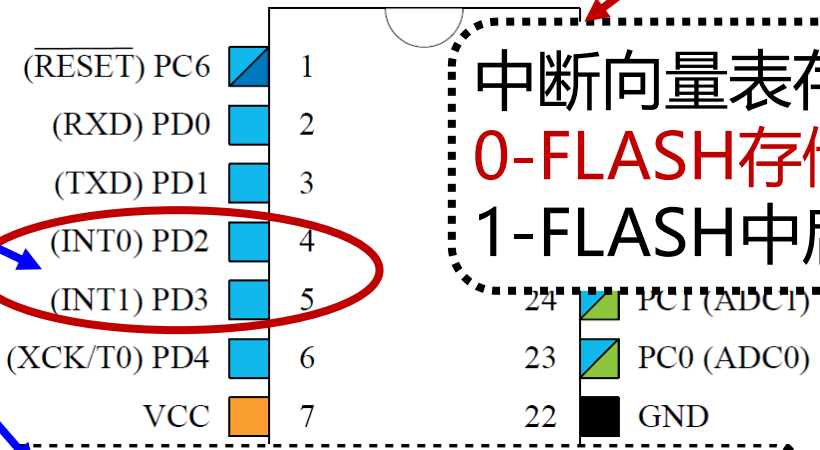
- 不同的中断源，入口地址不同
- 优先级
  - 中断号小的优先级高
  - 优先级高的可以打断优先级低的

中断号	入口地址	中断源	中断说明
1	0x000	RESET	外部，上电，掉电或看门狗复位
2	0x001	INT0	外部中断请求0
3	0x002	INT1	外部中断请求1
4	0x003	TIMER2 COMP	定时器/计数器2 比较匹配
5	0x004	TIMER2 OVF	定时器/计数器2 溢出
6	0x005	TIMER1 CAPT	定时器/计数器1 捕获事件
7	0x006	TIMER1 COMPA	定时器/计数器1 比较匹配A
8	0x007	TIMER1 COMPB	定时器/计数器1 比较匹配A
9	0x008	TIMER1 OVF	定时器/计数器1 溢出
10	0x009	TIMER0 OVF	定时器/计数器0 溢出
11	0x00A	SPI,STC	串行传输结束
12	0x00B	USART,RXC	USART，Rx接收结束
13	0x00C	USART,UDRE	USART数据寄存器空
14	0x00D	USART,TXC	USART，Tx发送结束
15	0x00E	ADC	ADC转换结束
16	0x00F	EE_RDY	EEPROM准备好
17	0x010	ANA_COMP	模拟比较器
18	0x011	TWI	两线串行接口
19	0x012	SPM_RDY	保存程序存储器准备好

ATmega8A外部中断的开或关

■通用中断控制寄存器**GICR**，IO空间地址：0x3B

Bit	7	6	5	4	3	2	1	0
GICR	INT1	INT0					IVSEL	IVCE
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0



中断向量表存在flash中的位置：
0-FLASH存储器的开始:0x000
1-FLASH中启动区的开始

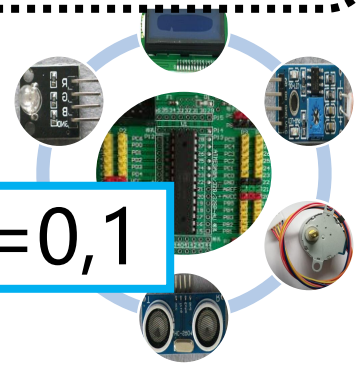
0:关中断　1:开中断

GICR &= ~(1<<INTn)

n=0,1

GICR |=(1<<INTn)

n=0,1

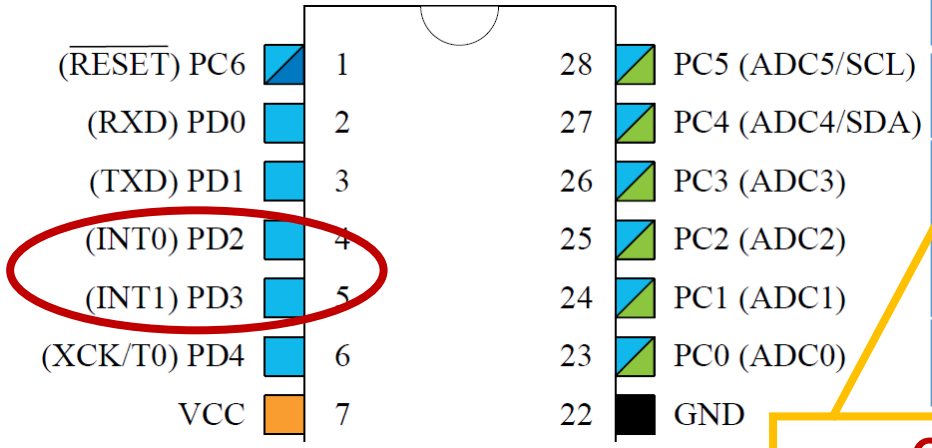


ATmega8A外部中断的触发控制

■MCU控制寄存器**MCUCR**，IO空间地址：0x35

Bit	7	6	5	4	3	2	1	0
MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SE、SMx开启/选择睡眠模式



ISCn1	ISCn0	说明
0	0	INTn低电平触发
0	1	INTn任意电平变换触发
1	0	INTn下降沿触发
1	1	INTn上升沿触发

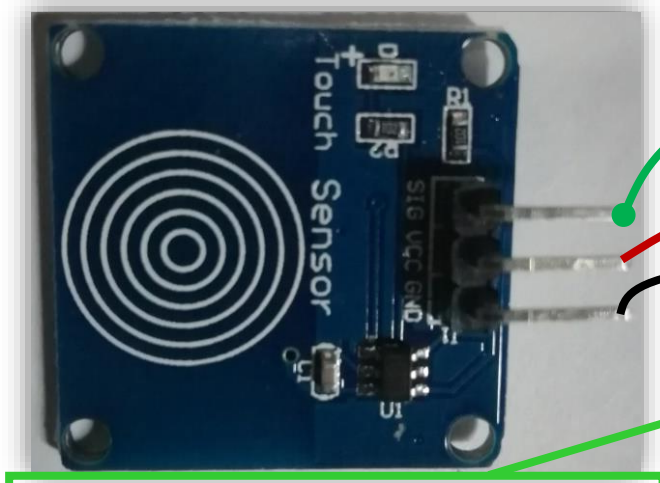
n=0,1

MCUCR =或&=或|=...

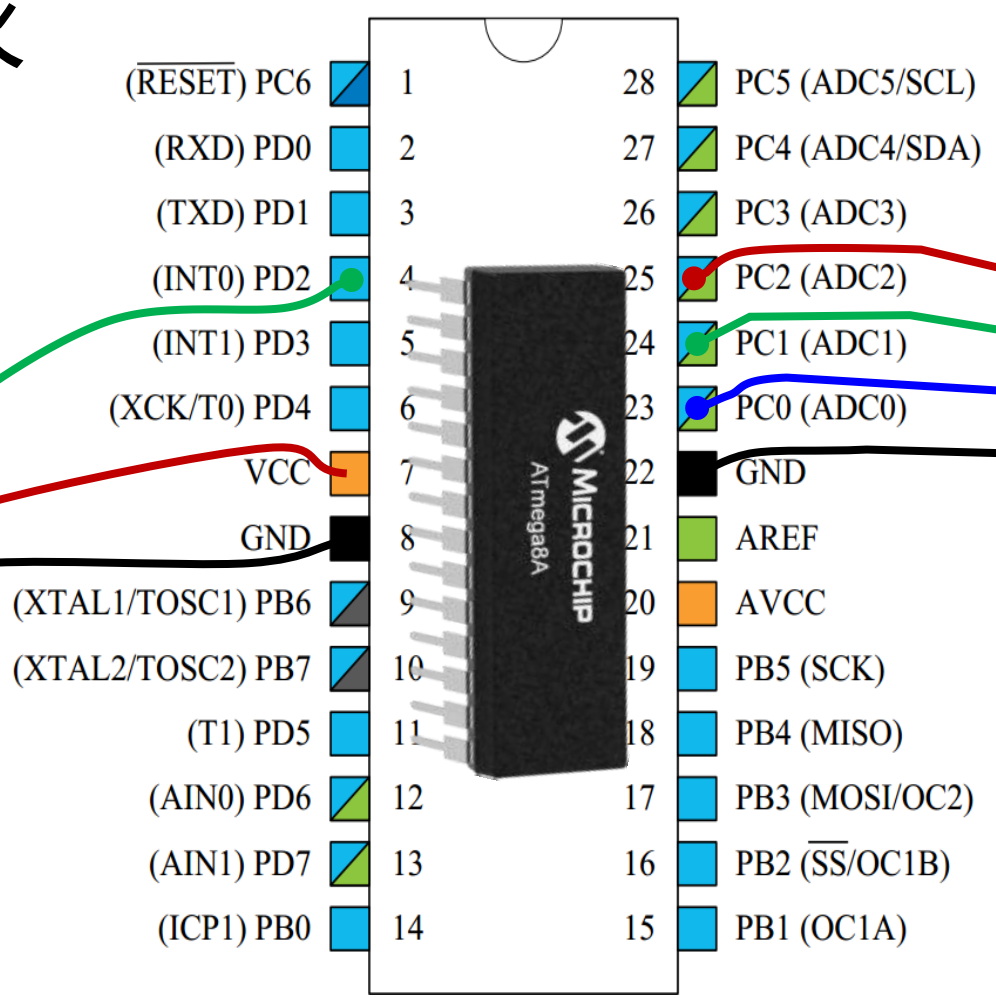
ATmega8A的中断编程实例

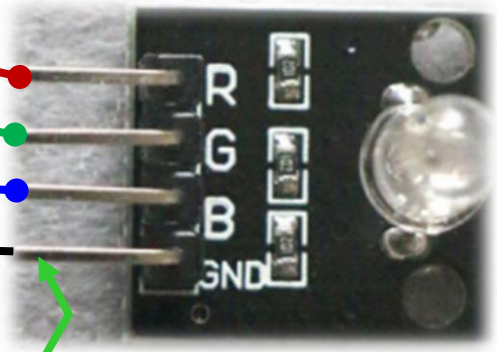
■用中断方式统计一位触摸开关的开/关次数，并控制RGB LED的亮/灭

VCC-电源正极，
GND-电源负极



就近连接到实验板上电源的正、负极





就近连接到实验板上电源的负极

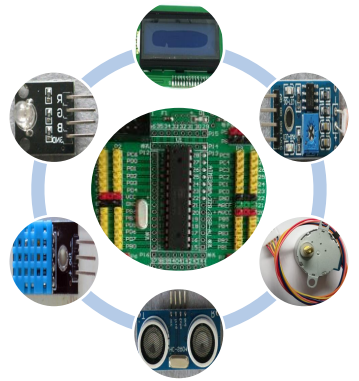
ATmega8A的**中断编程实例**：主函数/全局

```
#include <avr/io.h>
#include <avr/interrupt.h>
unsigned char counter=0;
int main(void)
{ DDRC = (1<<DDRC2)|(1<<DDRC1)|(1<<DDRC0);//PC2~0为输出
  DDRD &= ~(1<<DDRD2);//PD2(int0)为输入(来自tp223的sig)
  MCUCR |=((1<<ISC01)|(1<<ISC00));//int0上升沿触发中断
  GICR |= (1<<INT0);//允许INT0外部中断
  sei(); //开启全局中断SREG(I)=1
  while (1) {    switch(counter){
                        case 1 : PORTC = (1<<PORTC2);break;
                        case 2 : PORTC = (1<<PORTC1);break;
                        case 3 : PORTC = (1<<PORTC0);break;
                        default :PORTC =0;
                } //switch结束
                    } //while结束
                                } //main函数结束
```

头文件、全局变量

设置

任务



ATmega8A的**中断编程实例**：中断服务过程

ISR-Interrupt Service Routine

```
ISR (INT0_vect)
{
    if(counter <4)
    counter++;
    else
    counter = 0;
}
```

预定义：每个中断服务过程都是以ISR开始

预定义：表示INT0中断在向量表中的入口地址，默认预定义都是以中断类型 + “_vect"

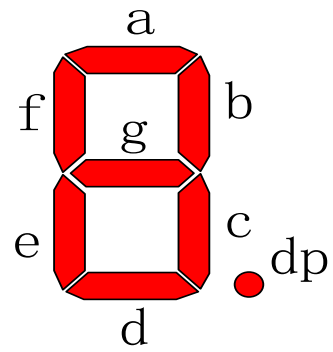
编译器：将中断服务过程的入口地址与中断向量表中对应中断源的跳转语句关联起来

INT0管脚出现上升沿脉冲时执行一次此服务过程

七段数码管：1位

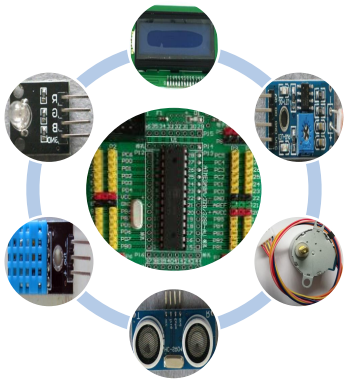
可用MCU的IO端口直接控制

10-pin

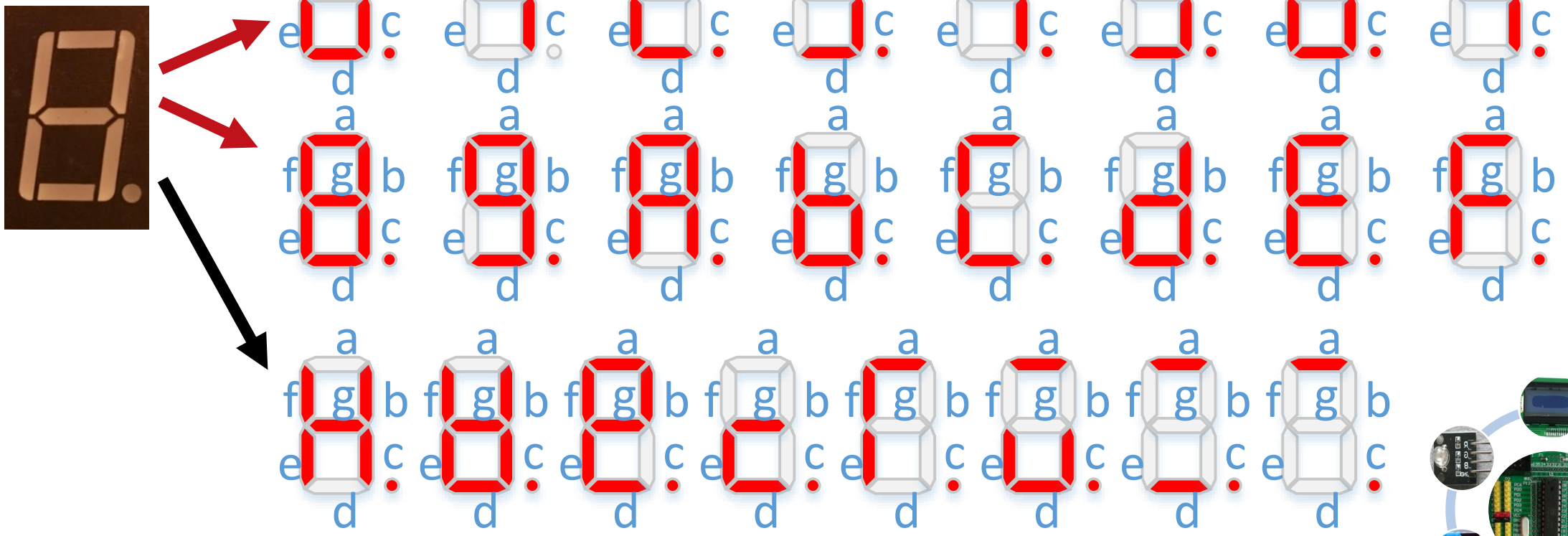


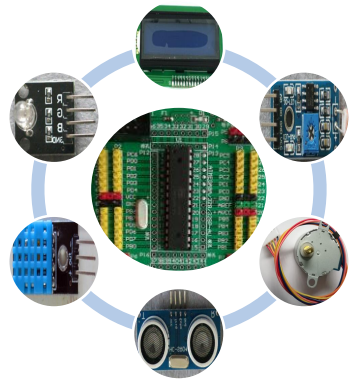
CC-Common Cathode

~~CA-Common Anode~~



七段数码管的数字显示（编/译码）

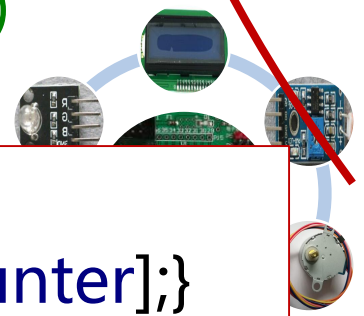




1位七段数码管的中断编程示例

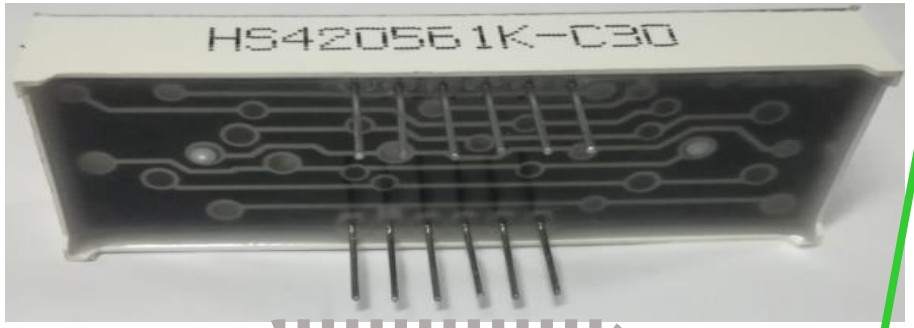
```
#include <avr/io.h>
#include <avr/interrupt.h>
unsigned char counter=0; //全局,开关次数
int main(void)
{  unsigned char seg7_hex[16]=
   {0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,
    0xfe,0xf6,0xee,0x3e,0x9c,0x7a,0x9e,0x8e};//MSB-a,b,...,dp-LSB
    DDRB = (0xff);//PB0~7为输出控制七段数码管的a~q,dp
    DDRC = (0x01);//PC0为输出控制数码管的共阴极管脚(如'-1')
    PORTC = (0x00);//PC0输出低电平 '0'
    DDRD &= ~(1<<DDRD2);//PD2(int0)为输入（从tp223触摸开关）
    MCUCR |=((1<<ISC01)|(1<<ISC00));//int0 上升沿触发中断
    GICR |= (1<<INT0);//允许INT0中断
    sei(); //开启全局中断SREG(I)
```

```
ISR(INT0 vect)//似函数
{   if(counter <4)
        counter++;
    else counter = 0;
}
```

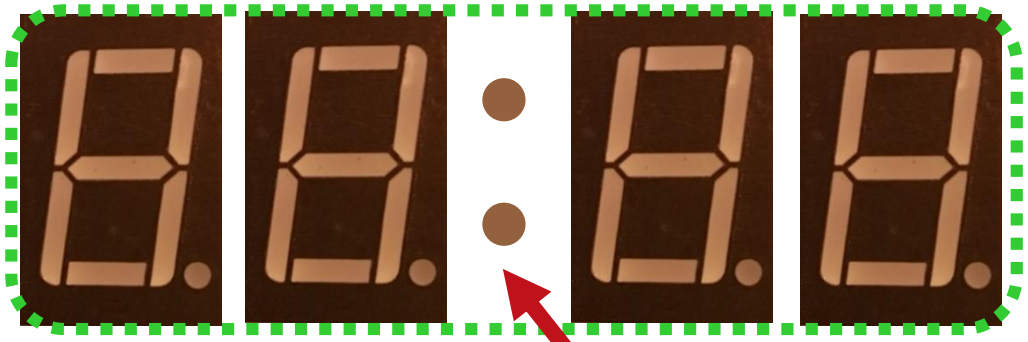


```
while (1)
    { PORTB = seg7_hex[counter];}
} //main函数结束
```

4位共阴极七段数码管的外形和管脚分布



12-pin

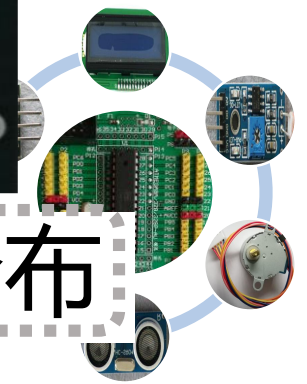


时钟

管脚编号

管脚分布

管脚名：-1~-4为4**位**共阴极，a~f,dp为**段**的正极



4位共阴极七段数码管的内部电路结构

4位共阴极七段数码管的工作方式1

dp
g
b
a

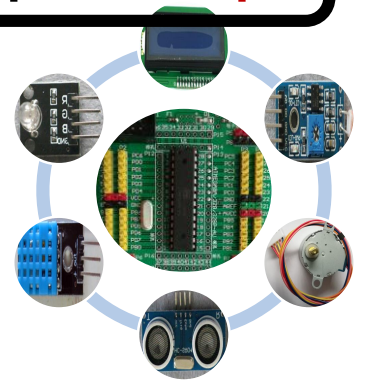
a
f g b
e c
d

b,c为1，
其它为0

0

-1
-2
-3
-4

⇨ 4位~1位



中国科学技术大学
University of Science and Technology of China

4位共阴极七段数码管的工作方式2

dp
g
b
a

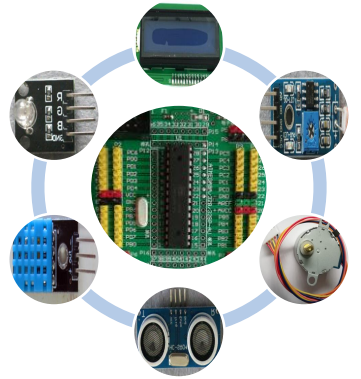
T1 T3 T2 T4

a f g b e c d

-1 -2 -3 -4

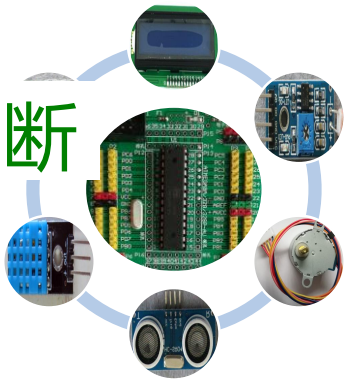
T1 T2 T3 T4

⇨ 4位~4位



4位共阴极数码管编程示例(1)

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
unsigned int counter=0; //全局变量
int main(void)
{ unsigned char seg7_hex[16]=
  {0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,
   0xfe,0xf6,0xee,0x3e,0x9c,0x7a,0x9e,0x8e};//MSB-a,...,g,dp-LSB
   unsigned char i,seg7_com[4]={0xe,0xd,0x0b,0x07};
   unsigned int d_t;
   DDRC = (0x0f);//PC3~0为输出，控制公共端-1,-2,-3,-4
   DDRB = (0xff);//PB7~0为输出，对应控制a,b,...,g,dp段的正极
   DDRD &= ~(1<<DDRD2);//PD2(int0)连接触摸开关
   MCUCR |=((1<<ISC01)|(1<<ISC00));//int0 上升沿触发中断
   GICR |= (1<<INT0);//允许INT0外部中断
   sei(); //开启全局中断SREG(I)
```

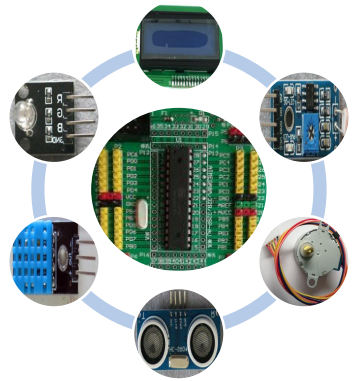


4位共阴极数码管编程示例(2)

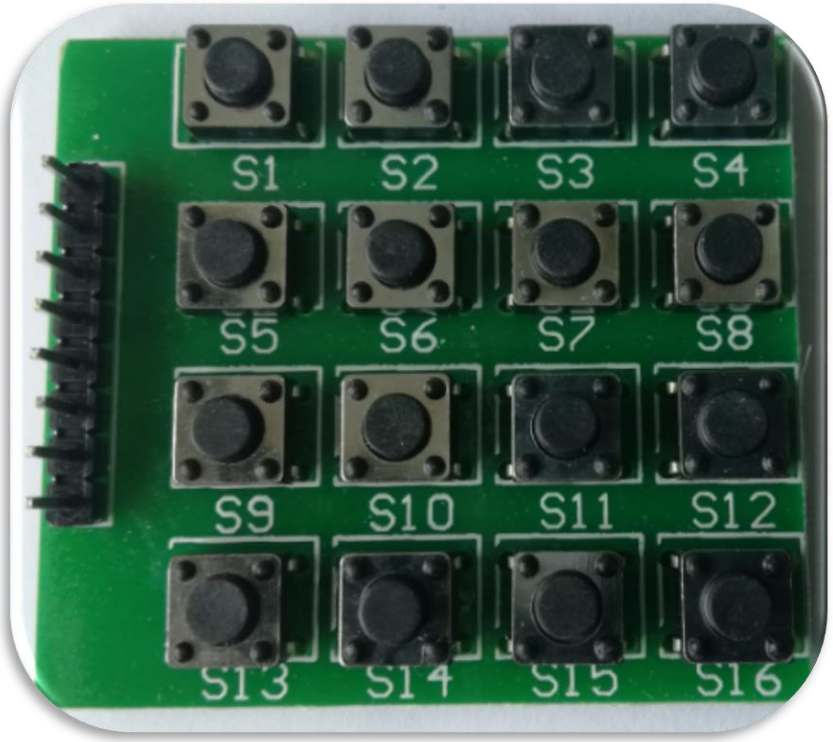
```
while (1)
{  d_t = counter;
    for(i=0;i<4;i++)
    {  PORTC |=0x0f; //禁止显示
        PORTB =seg7_hex[d_t%10]; //
        给数
        PORTC =seg7_com[i]; //显示
        d_t=d_t/10; //下一位
        _delay_ms(5);
    }
  }
} //main函数结束
```

示例中数码管等与MCU的连接			
数码管	**MCU**	**数码管**	**MCU**
a	PB7	g	PB1
b	PB6	dp	PB0
c	PB5	-1	PC3
d	PB4	-2	PC2
e	PB3	-3	PC1
f	PB2	-4	PC0
触摸开关SIG		**PD2**	

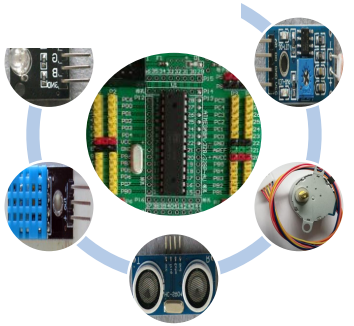
```
ISR(INT0_vect)
{if(counter < 9999) counter++;
 else   counter = 0; }
```



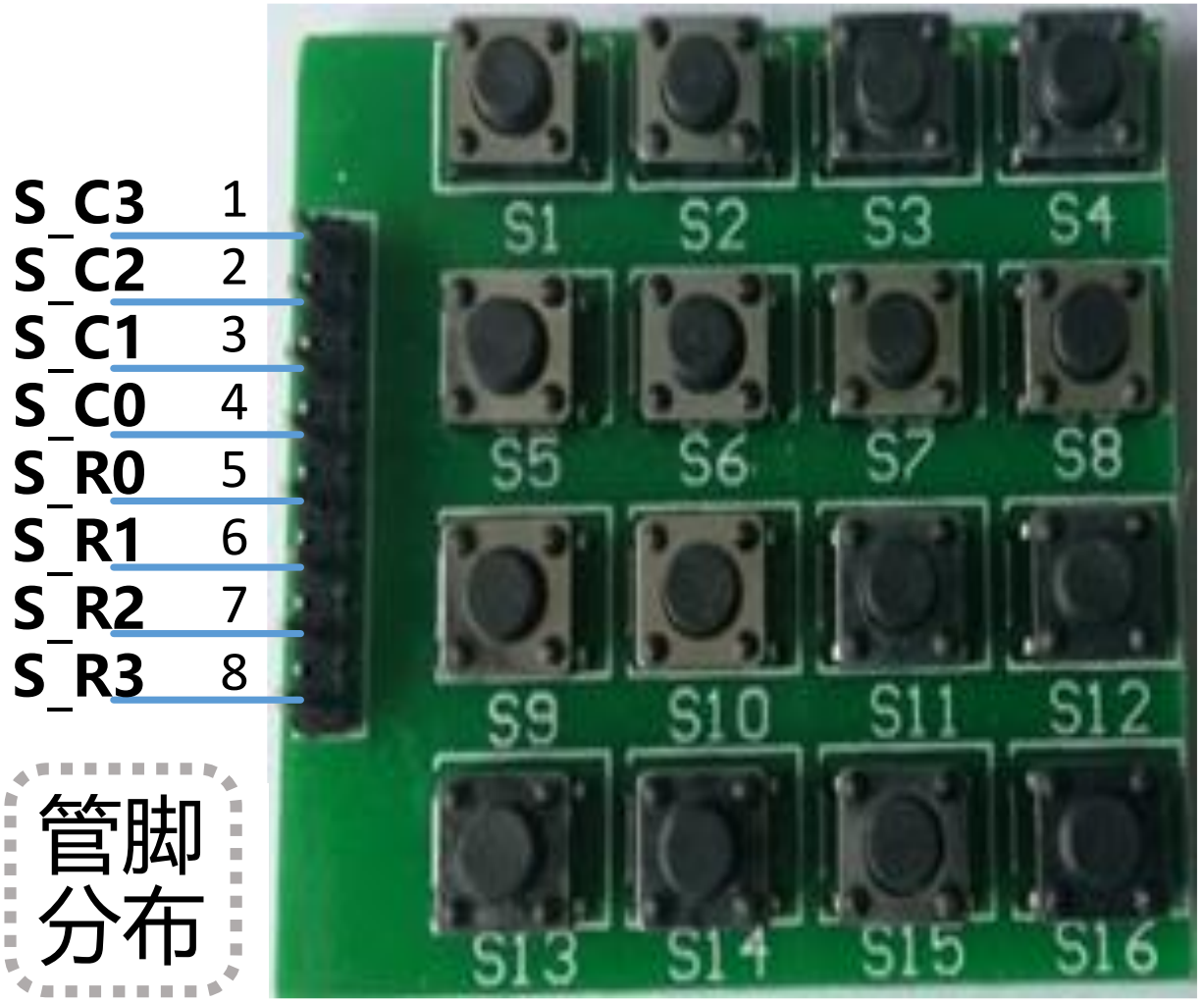
4×4按键阵列


S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16

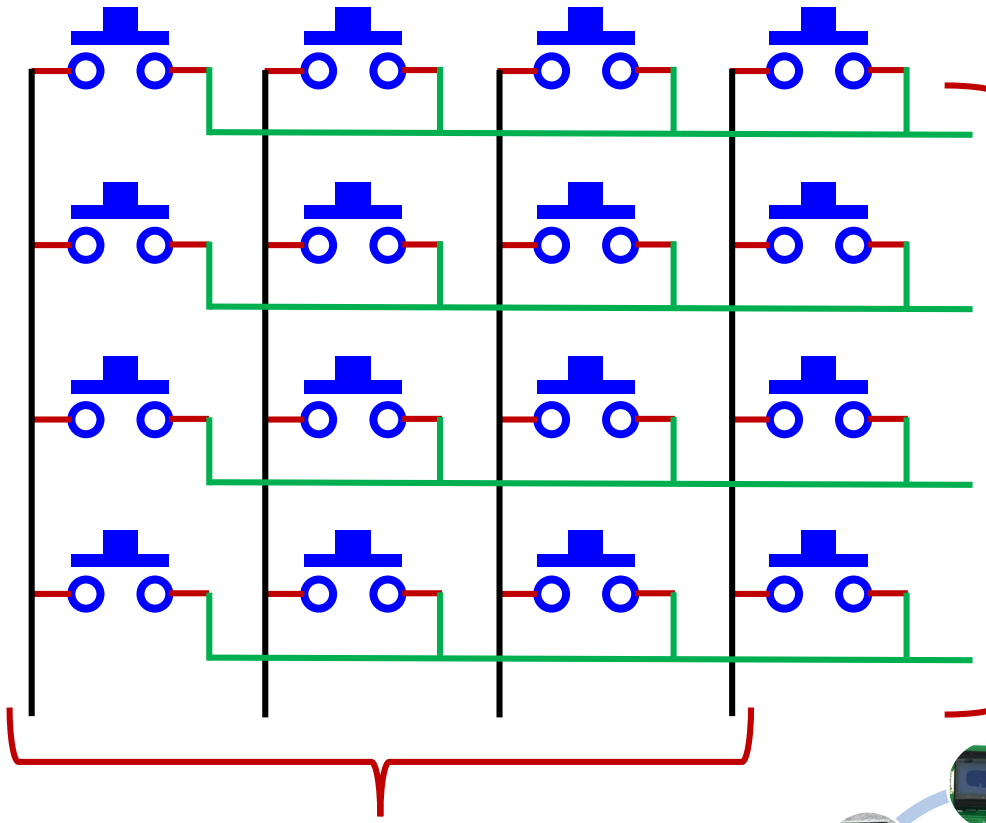

A
B
C
D
3
6
9
#
2
5
8
0
1
4
7
*

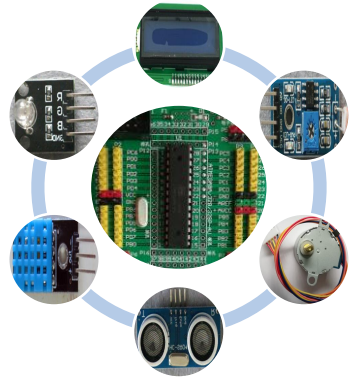


4×4按键阵列1的结构与管脚分布

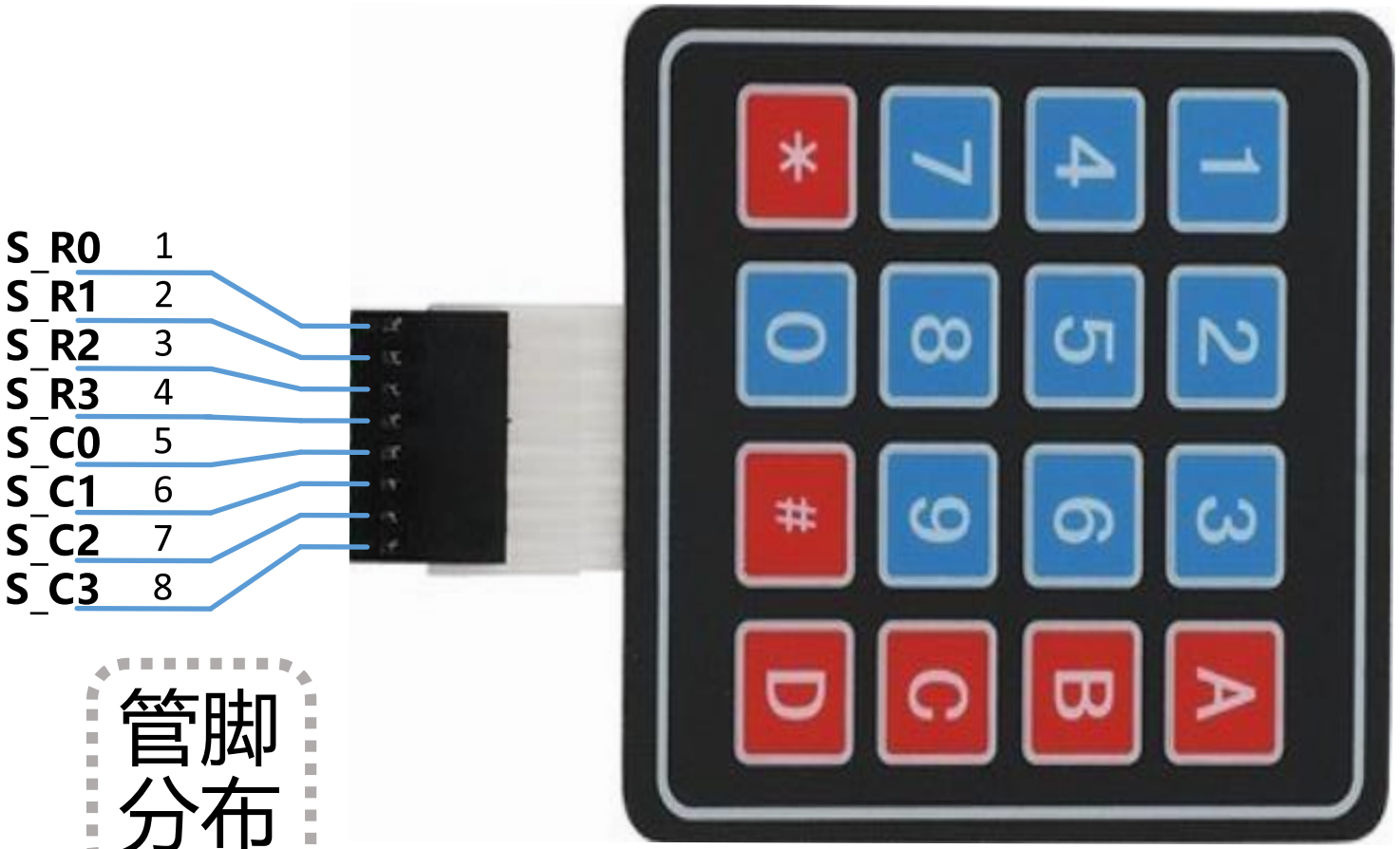


管脚分布

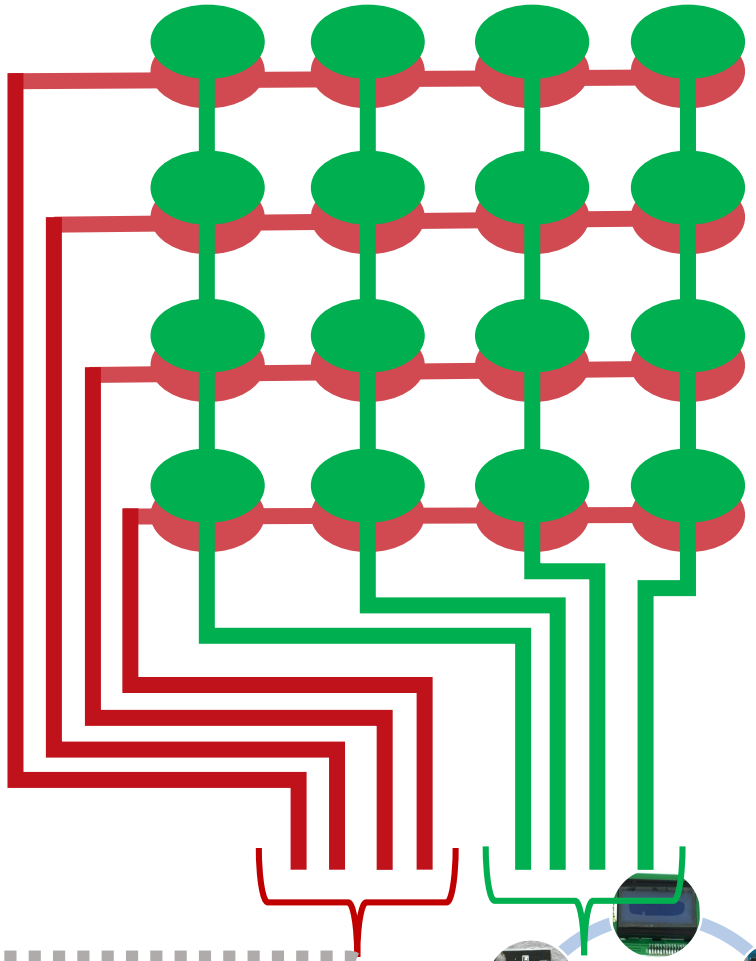




4×4按键阵列2的结构与管脚分布



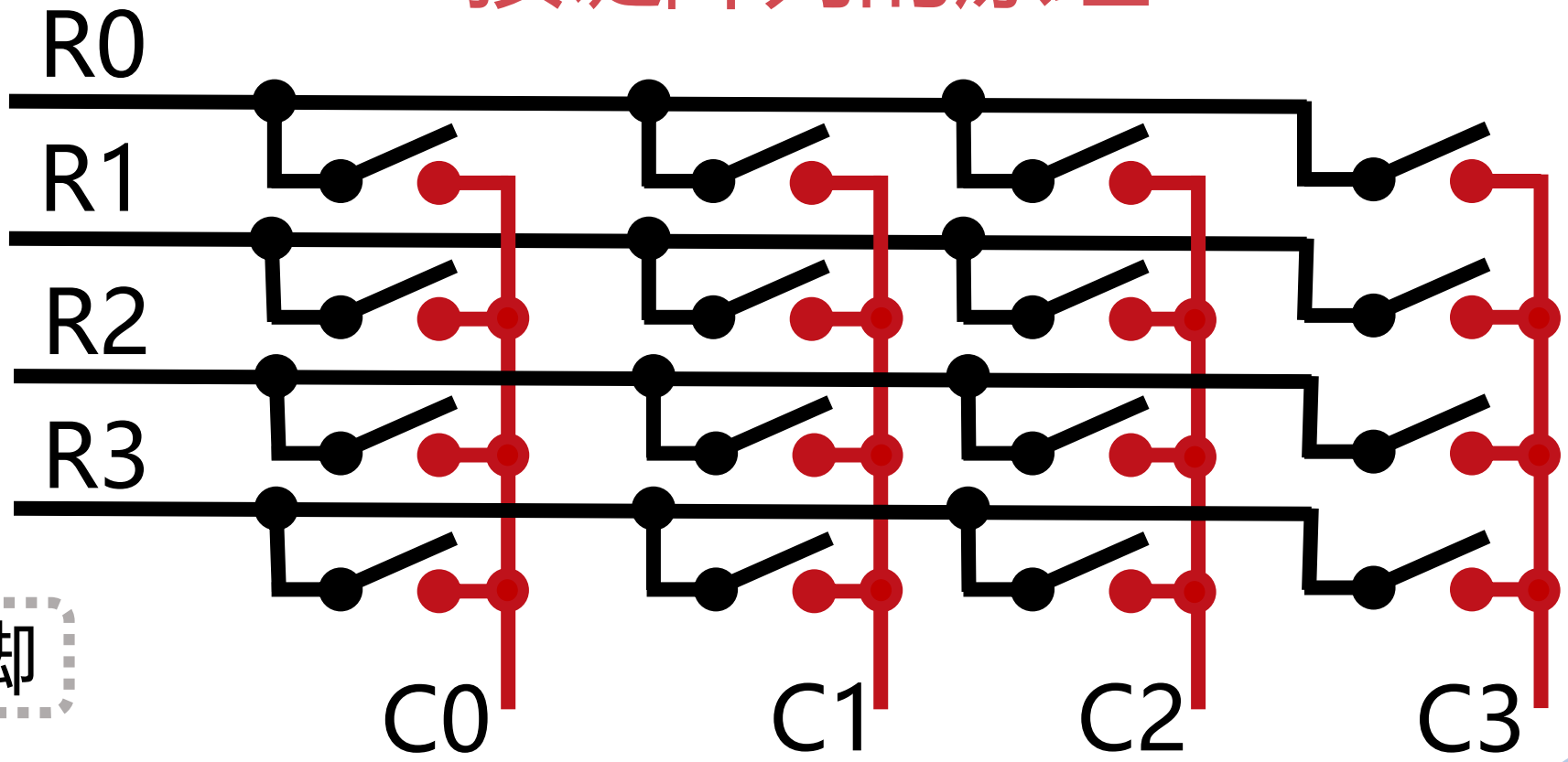
管脚分布



4个行管脚

4个列管脚

4×4按键阵列的原理

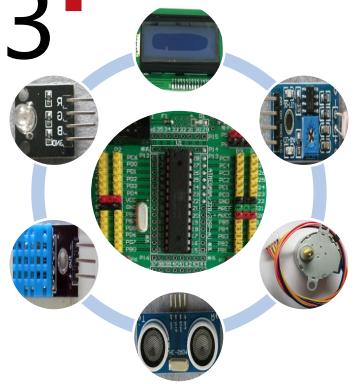


行控制线

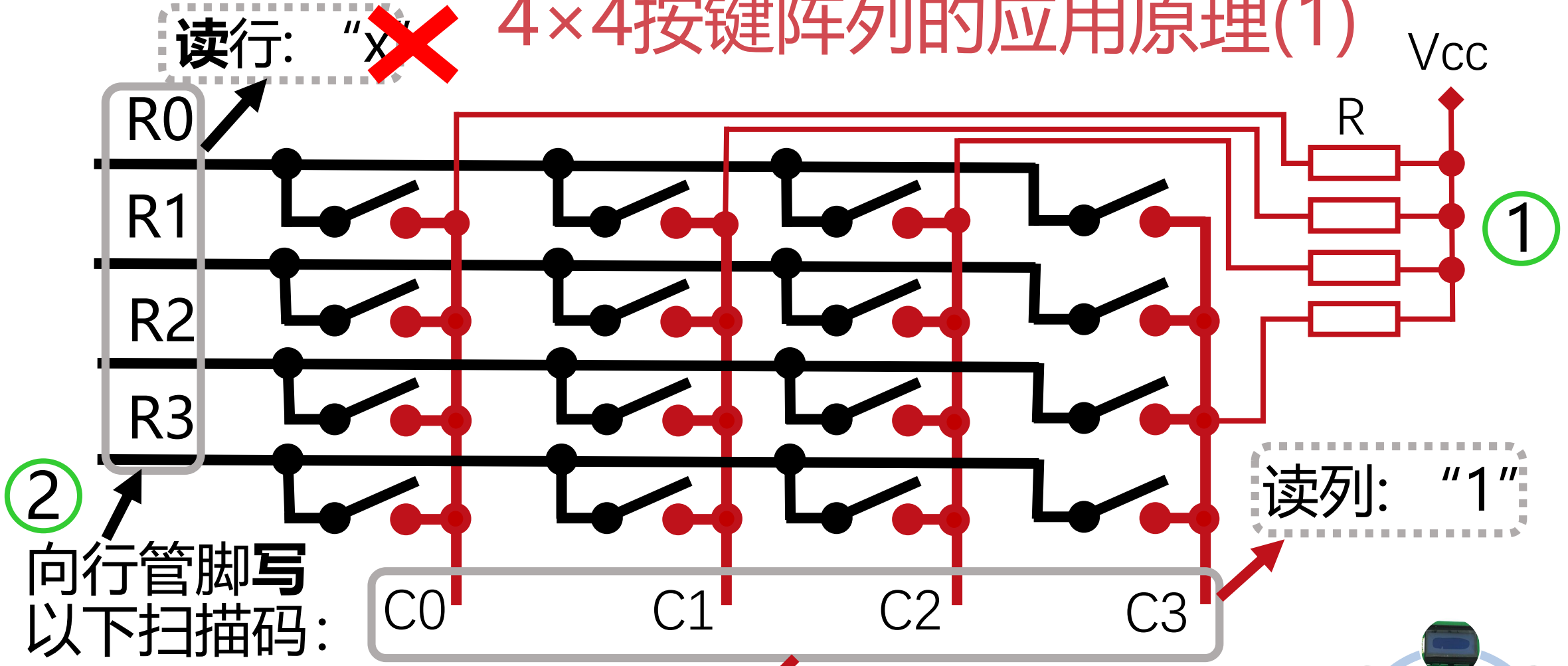
4个行管脚

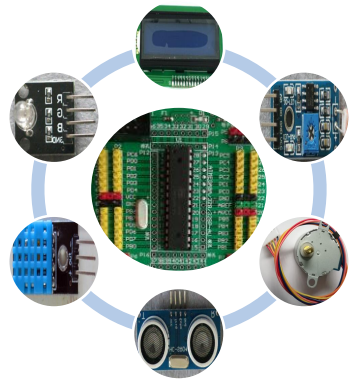
列控制线

4个列管脚

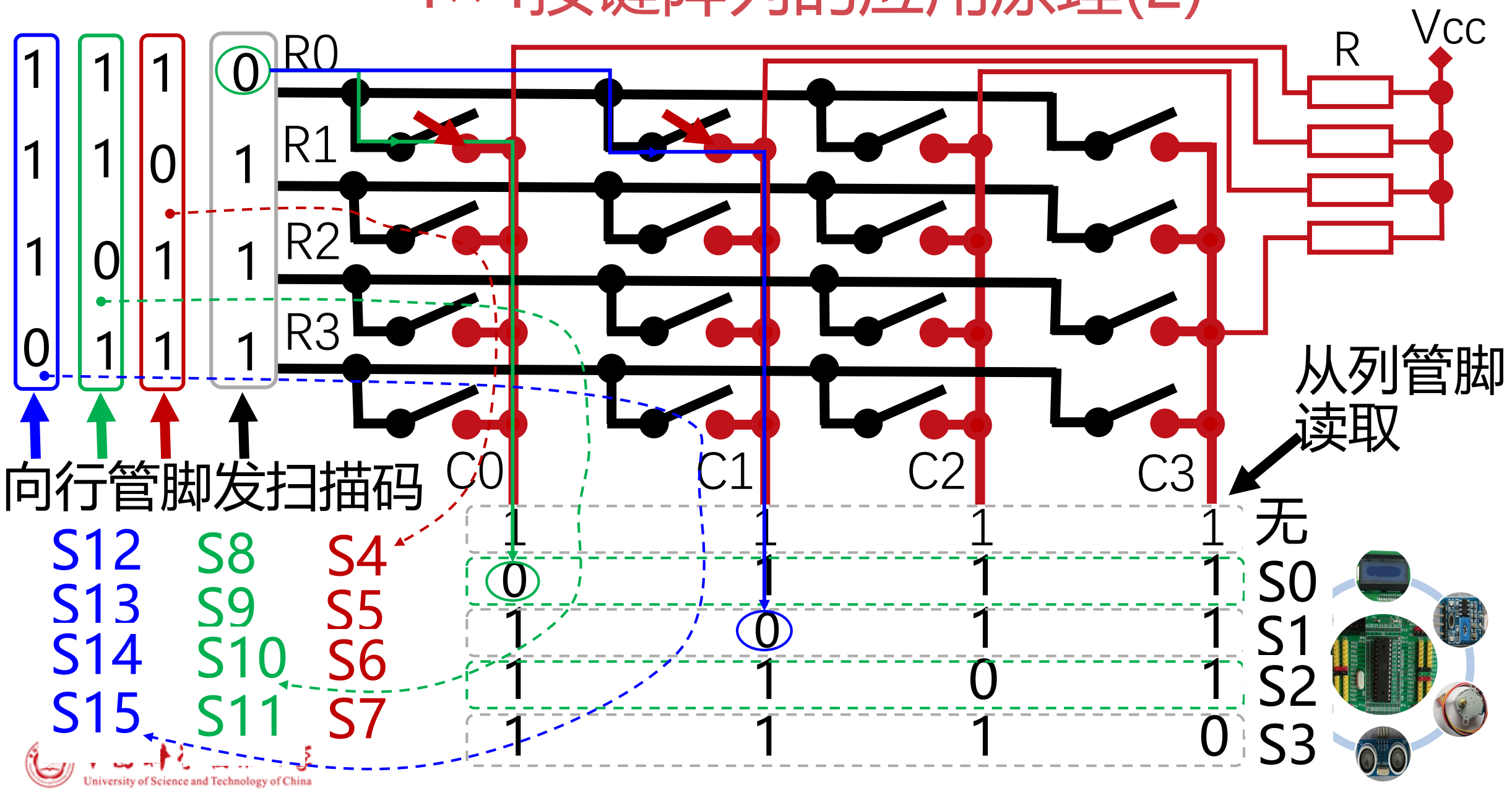


4×4按键阵列的应用原理(1)

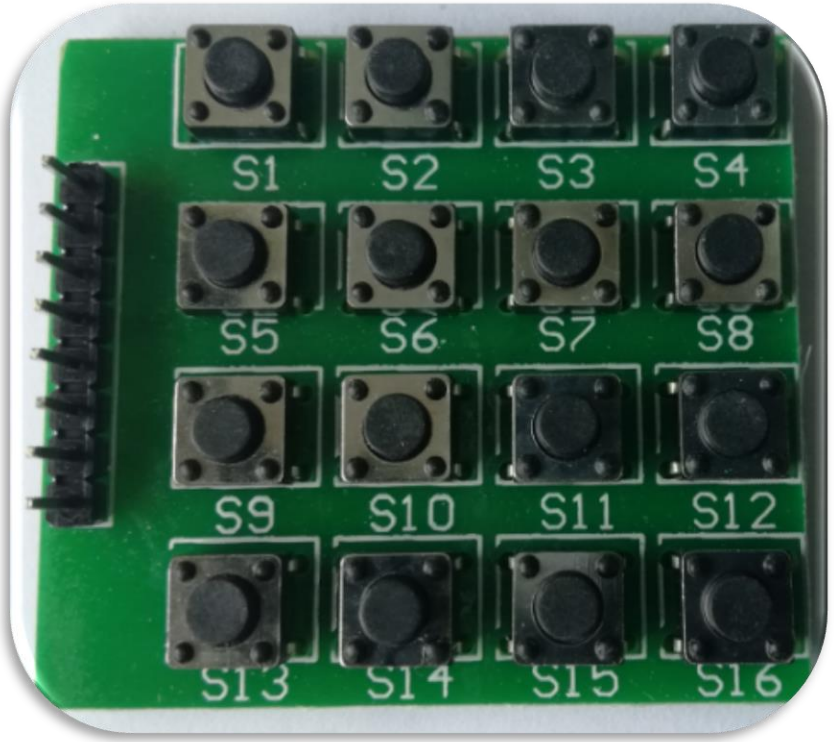




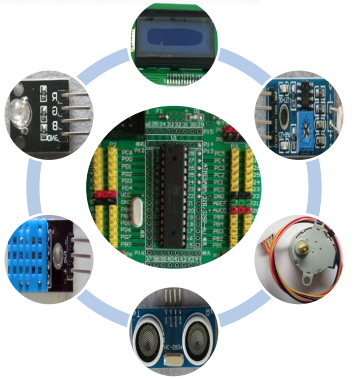
4×4按键阵列的应用原理(2)



4×4按键阵列的编程示例



将小键盘最近四次按下的键，按顺序编码并显示在数码管上

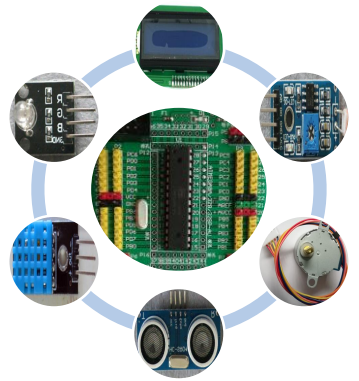


4×4按键阵列的编程示例代码（1）

```
#include <avr/io.h>
int main(void)
{ unsigned char
  seg7_hex[17]={0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,
    0xfe,0xf6,0xee,0x3e,0x9c,0x7a,0x9e,0x8e,0x02};//4-7译码
  unsigned char i,seg7_com[4]={0xe,0xd,0x0b,0x07};//扫描码
  unsigned char seg7_no[4]={16,16,16,16};//要显示的4位数字
  unsigned char getkey,keyno;//取扫描输入，对按下按键的编码
  DDRC = (0x0f);//PC3~0输出到公共端-1，-2，-3，-4
  DDRB = (0xff);//PB7~0输出到a,b,...,g,dp段的正极
  DDRD = (0x0f);//经PD3~0输出扫描码到行(S_R3~0)，从列(S_C3~0)
输入到PD7~4
PORTD = (0xff);//PD7~4的内部上拉电阻启用,并PD3~0输出高电平
#define F_CPU 1000000UL
#include <util/delay.h>
```

4×4按键阵列的编程示例代码（2）

```
while (1)
{   /*按键阵列扫描，判断，编码*/
    keyno = 16;//默认无按键按下
    //1.扫描第1行
    PORTD = ~(1<<PORTD0);//PORTD0为低电平，扫描第1行
     delay us(2);
    getkey = (PIND & 0xf0)>>4;//获取列状态,并移动到低4位
    switch(getkey)
    { case 0x0e:keyno = 0;break;//1行1列为S0
      case 0x0d:keyno = 1;break;//1行2列为S1
      case 0x0b:keyno = 2;break;//1行3列为S2
      case 0x07:keyno = 3;break;//1行4列为S3
    }
```



4×4按键阵列的编程示例代码（3）

```
//2.扫描第2行
PORTD = ~(1<<PORTD1);//PORTD1为低电平，扫描第2行
  delay us(2);
getkey = (PIND & 0xf0)>>4;//获取列状态,并移动到低4位
switch(getkey)
{ case 0x0e:keyno = 4;break;
   //2行1列为S4
  case 0x0d:keyno = 5;break;
   //2行2列为S5
  case 0x0b:keyno = 6;break;
   //2行3列为S6
  case 0x07:keyno = 7;break;
   //2行4列为S7
}
```

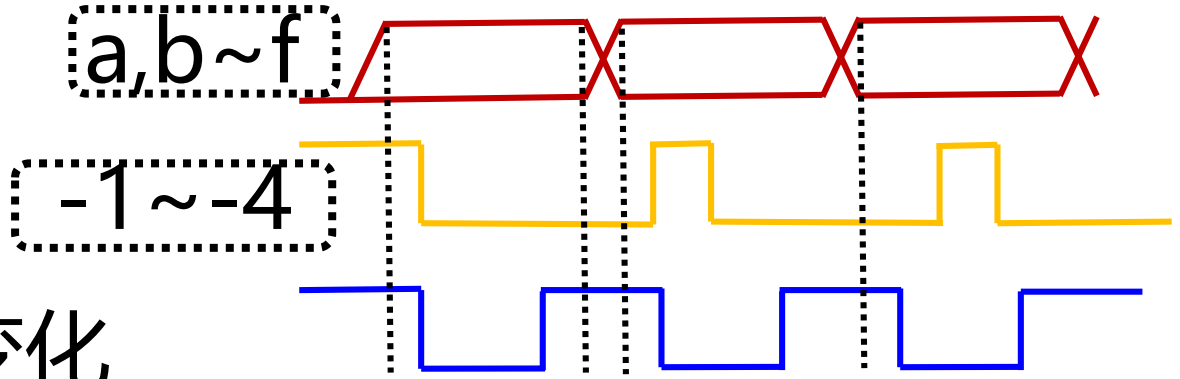
```
//3.扫描第3行
PORTD = ~(1<<PORTD2);//扫描第3行
  delay us(2);
getkey = (PIND & 0xf0)>>4;//获取列状态
switch(getkey)
{ case 0x0e:keyno = 8;break;//3行1列为S8
  case 0x0d:keyno = 9;break;//3行2列为S9
  case 0x0b:keyno = 10;break;//...为S10
  case 0x07:keyno = 11;break;//...为S11
}
```

4×4按键阵列的编程示例代码4

```
//4.扫描第4行
PORTD = ~(1<<PORTD3);//PORTD3为低电平，扫描第4行
  delay_us(2);
getkey = (PIND & 0xf0)>>4;//获取列状态,并移动到低4位
switch(getkey)
{ case 0x0e:keyno =12;break;//4行1列为S12
  case 0x0d:keyno =13;break;//4行2列为S13
  case 0x0b:keyno =14;break;//4行3列为S14
  case 0x07:keyno =15;break;//4行4列为S15
}
if(keyno<16) /*扫描一轮并编码后*/
{ seg7_no[3]=seg7_no[2]; //移动按键数据
  seg7_no[2]=seg7_no[1];
  seg7_no[1]=seg7_no[0];
  seg7_no[0]=keyno; //新的按键数据
}
```

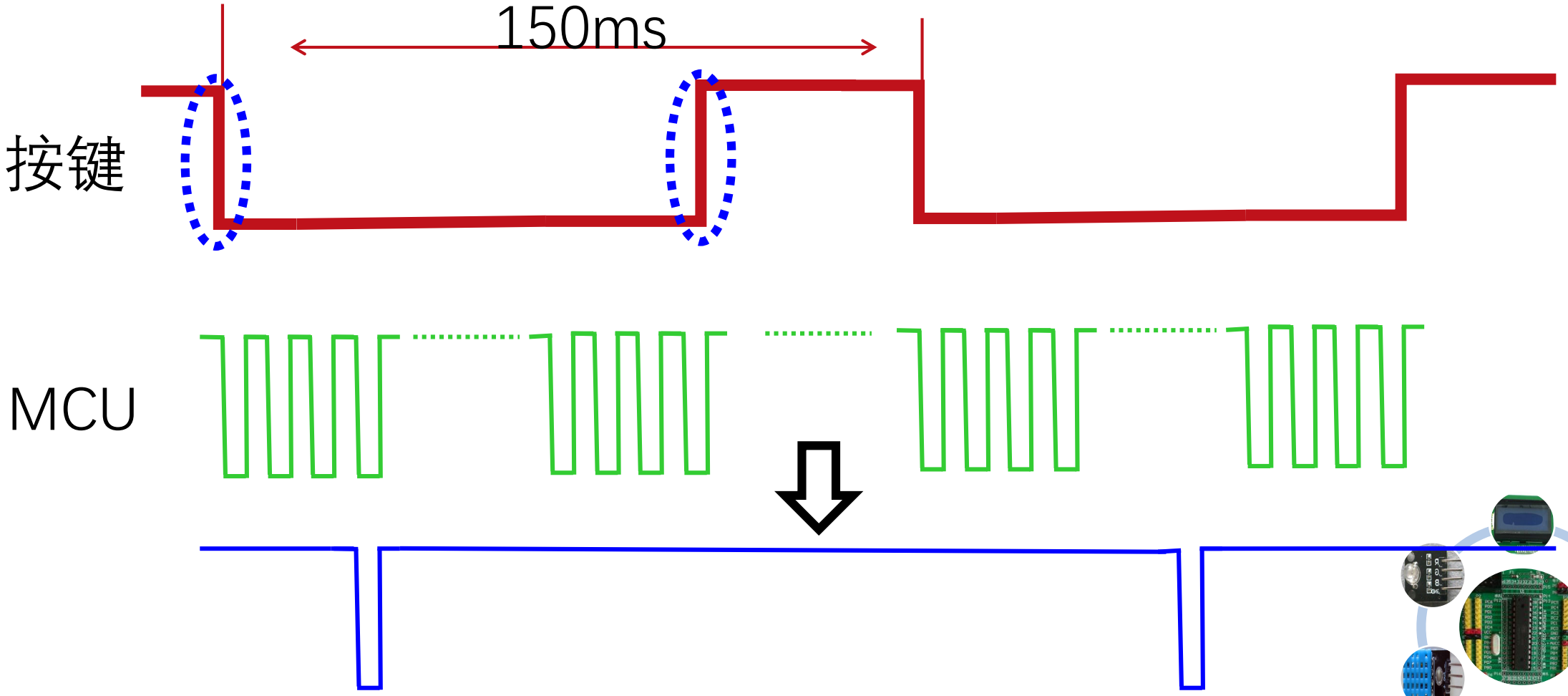
```
/*4位七段数码管动态扫描显示*/
 for(i=0;i<4;i++)
 {
  PORTC |= 0x0f; //禁止显示
  PORTB = seg7_hex[seg7_no[i]];
  PORTC = seg7_com[i]; //显示
   delay_ms(1);
 }
} //while结束
} //main函数结束
```

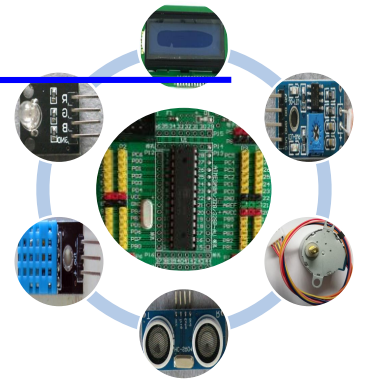
按键阵列与共阴极七段数码管知识拓展



- 七段数码管动态描显示会出现叠加现象
  - 避免数码管选中时出现数据变化
- 按键阵列输入时会出现MCU处理时间与按键按下时间不匹配的情况，即显示重复输入现象
  - 降低MCU处理时间：统计、延时
  - 改进MCU处理方式：边沿、中断

按键阵列与七段数码管知识拓展





本周实验内容

实验内容1：将**INT1**管脚上连接的触摸开关的开关次数显示在一位数码管上(显示在4位七段数码管的任一位即可，参见PPT“4位共阴极七段数码管的外形和管脚分布”等)

实验内容2：将**INT1**管脚上的触摸开关的开关**次数**显示在**四位**共阴极七段数码管上

实验内容3：根据小键盘示例和拓展知识等，真正实现将小键盘**最近四次**按下的键，按**顺序**编码并**显示**在数码管上

当次全部完成后当场验收，总结下次交