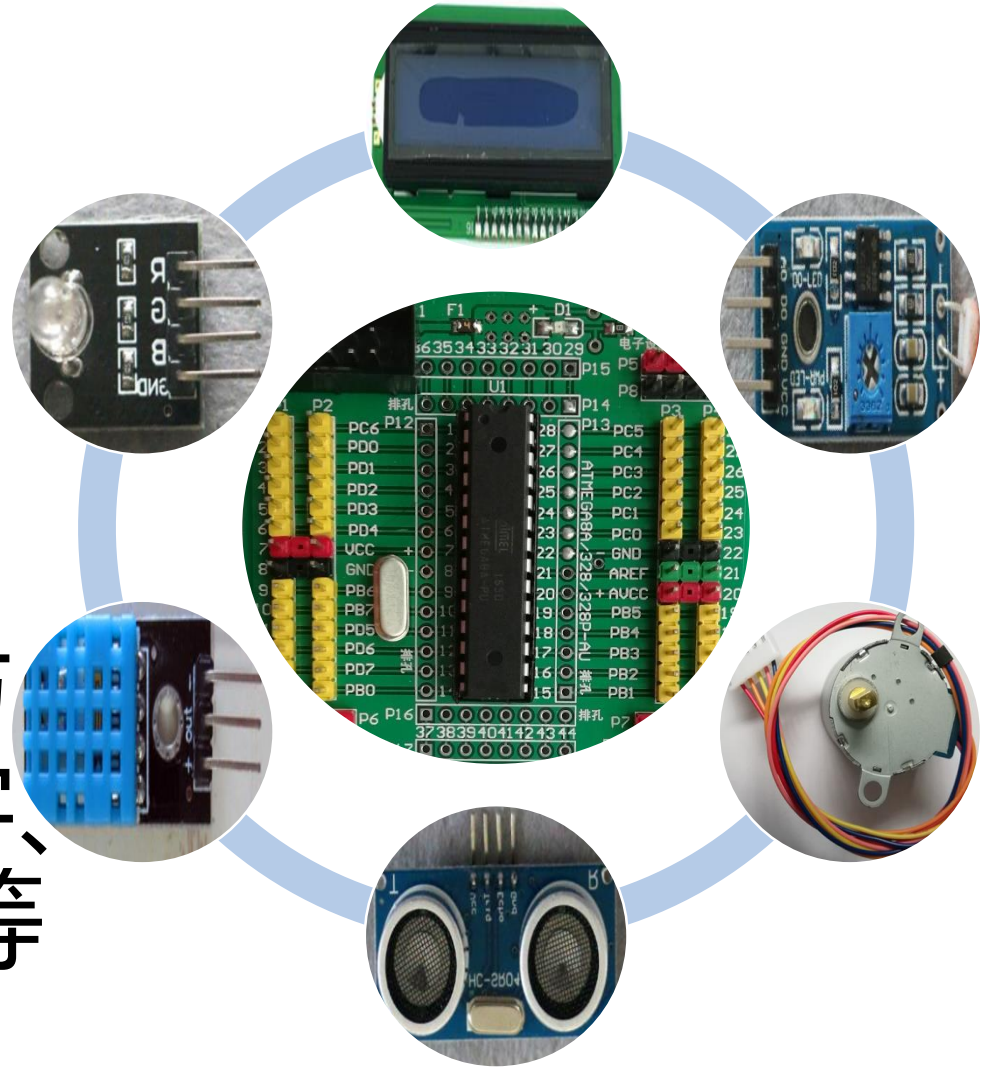
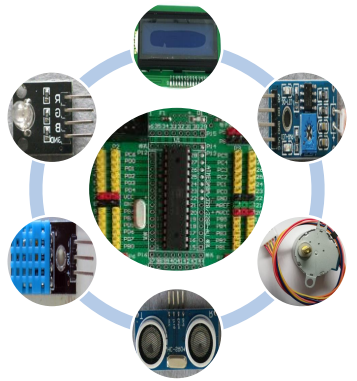
电子设计实践基础

MCU任务调度、中断与小键盘、LCD命令与汉字、实时时钟、蜂鸣器发声等



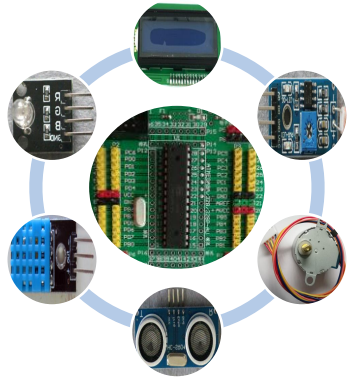
上节课内容回顾

- MCU定时器/计数器、 PWM及其程序编写
  - MCU定时器0,1,2
  - PWM（定时器1,2）
- 步进电机及其转动控制
- 直流电机及其转动控制



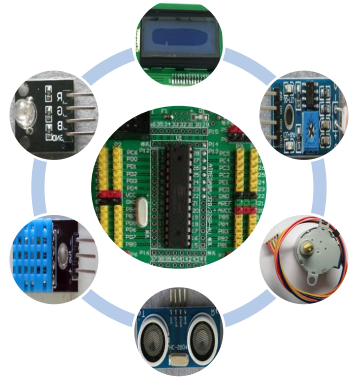
本节课主要内容

- MCU的“OS”（打破常规）
- 定时器中断与小键盘
- LCD命令与汉字（自定义字符）
- 实时时钟模块
- 蜂鸣器发声



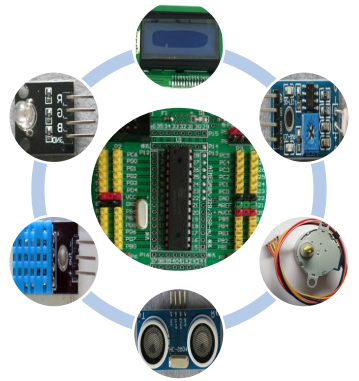
MCU的“OS”— 打破常规

- MCU的“OS”：别逗了
- 避免MCU大量的长延时(空转)，提高MCU的利用率
- 任务调度：时间片、轮转、多个任务、调度（优先、抢占...）；MMU、。。。



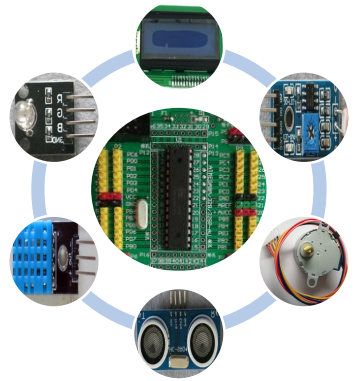
MCU的简单调度示例：（1）

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "twi  lcd.h"
#define MAX  TASKS 3
typedef struct {
        void(*ftask)(void);//指向函数的指针，即要执行的任务
        int ticks; //执行此任务的时间间隔
}OSTASK;  //任务结构类型，方便处理
unsigned char tp  cnt=0;//统计触摸开关的开关次数
void tp  task() //统计触摸开关的开关次数
{       static unsigned char tp1=0,tp2=0;//读取开关的状态,保留上一次的结果
        DDRB &= ~(1<<DDRB0);//PB0管脚连接触摸开关的开关信号
        tp2 = tp1;//缓存
        tp1 = (PINB & (1<<PINB0));//读取PB0管脚的状态
        if(tp2==0 && tp1==1)tp_cnt++; //统计开关次数
}
```



MCU的简单调度示例：（2）

```
void Rgb  show()
{       DDRC |=(1<<DDRC0)|(1<<DDRC1)|(1<<DDRC2);//为输出
        switch(tp  cnt)//根据触摸开关的开关次数点亮LED
        {       case 0: PORTC |=(1<<PORTC0);break;
                case 2: PORTC |=(1<<PORTC1);break;
                case 4: PORTC |=(1<<PORTC2);break;
                case 1: PORTC &=~(1<<PORTC0);break;
                case 3: PORTC &=~(1<<PORTC1);break;
                case 5: PORTC &=~(1<<PORTC2);break;
                case 6: PORTC |=(1<<PORTC0);break;
                case 7: PORTC |=(1<<PORTC1);break;
                case 8: PORTC |=(1<<PORTC2);break;
                case 11: PORTC &=~(1<<PORTC0);break;
                case 10: PORTC &=~(1<<PORTC1);break;
                case 9: PORTC &=~(1<<PORTC2);break;
                default:tp_cnt=0; break;
        }    }
```

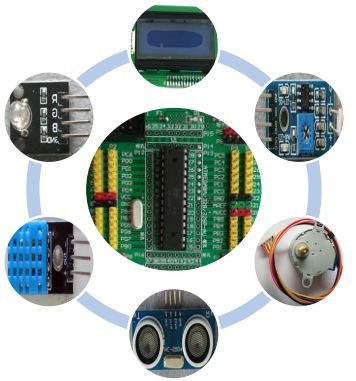


MCU的简单调度示例：（3）

```
void lcd_show()
{       char ch[5]={'0'},i;//
        unsigned char t=tp_cnt;
        for(i=0;i<3;i++)
        {       ch[2-i]=t%10+48;//数字转换为字符
                t/=10;//去掉低位
        }
        cli(); //全局中断关，不然会影响显示过程，定时器0中断服务也停止了
        LCD_Write_String(0,1,"times:");
        LCD_Write_String(0,8,ch);
        sei(); //全局中断开
}
int ticks[MAX_TASKS]={3,7,23}; //对应于任务执行间隔（时间片），单位1ms
OSTASK runtask[MAX_TASKS]={{tp_task,3},{Rgb_show,7},{lcd_show,19}};
//任务与执行时间操作列表，方便循环处理
```

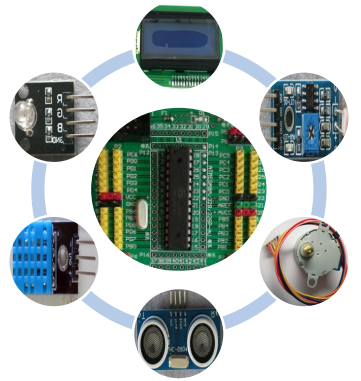
MCU的简单调度示例：（4）

```
int main(void)
{   unsigned char i=0;
    TWI_Init();          LCD_Init(); //lcd1602初始化
    //定时器0的初始化：1ms的定时与中断，用做时间片基准，中断定时等
    TCNT0 = 131; //1MHz的8分频，从131计数到255中断一次正好1ms
    TIMSK = (1<<TOIE0) ;//定时器0的溢出中断开
    TCCR0 = (1<<CS01); //CPU时钟的8分频，定时器0开始工作
    sei();//全局中断开
    while (1)
    {    for(i=0;i<MAX_TASKS;i++) //调度
         {       if(runtask[i].ticks==0)//到达此任务的执行（时间片耗尽）
                 {       runtask[i].ticks=ticks[i];//恢复此任务的时间片
                         runtask[i].ftask(); //执行相应的任务
                 }
         }
         for(i=0;i<100;i++); /*延时*/    } } //while和main结束
```



MCU的简单调度示例：（5）

```
ISR(TIMER0_OVF_vect)
{
        TCNT0 =131;//溢出后TC0从初始值重新计数
        for(unsigned char i=0;i<MAX  TASKS;i++)
                runtask[i].ticks--;  //时间片调整
}
```



ATmega8A的中断：中断向量表

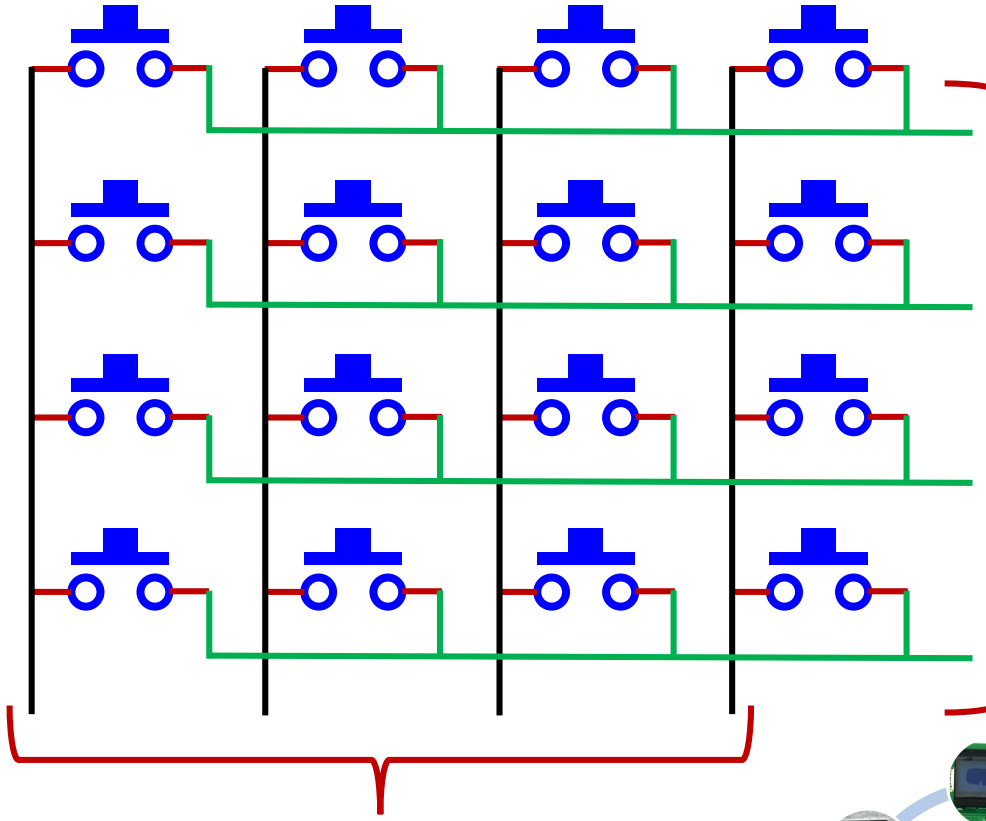
- 不同的中断源
- 不同的中断入口地址
- 优先级
  - （中断号小的优先级高）
  - 优先级高的可以打断优先级低的服务过程

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

4×4按键阵列1的结构与管脚分布

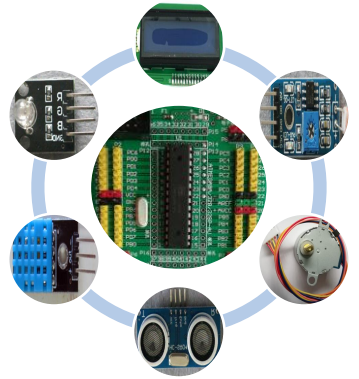
S_C3 1
S_C2 2
S_C1 3
S_C0 4
S_R0 5
S_R1 6
S_R2 7
S_R3 8

管脚分布

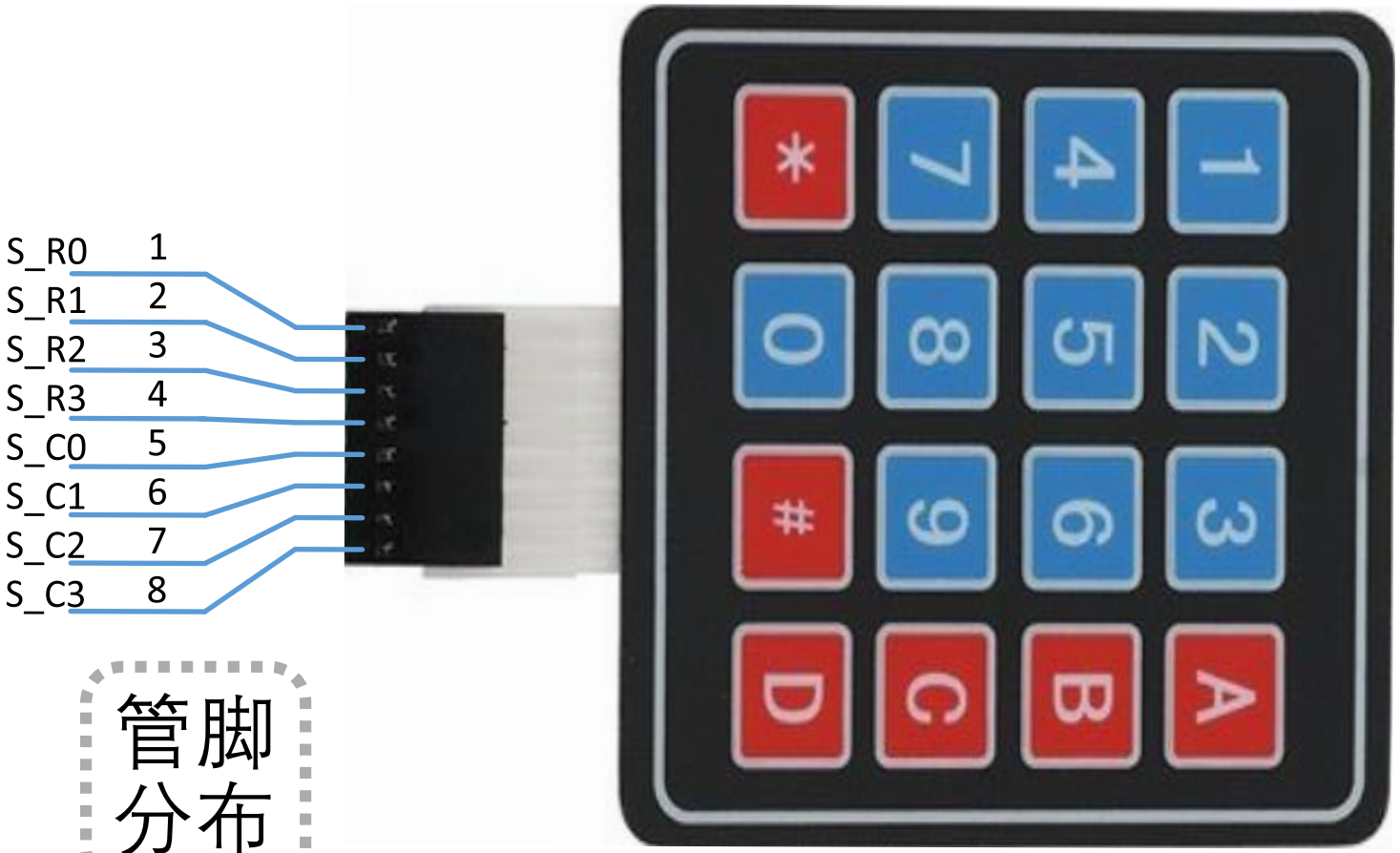


4个行管脚

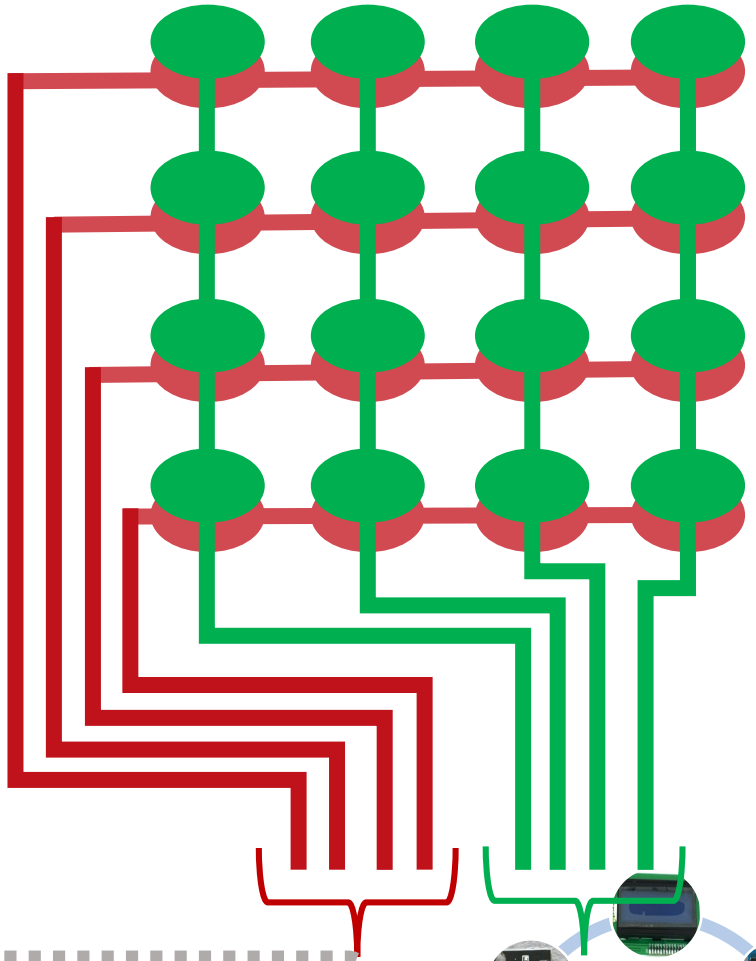
4个列管脚



4×4按键阵列2的结构与管脚分布



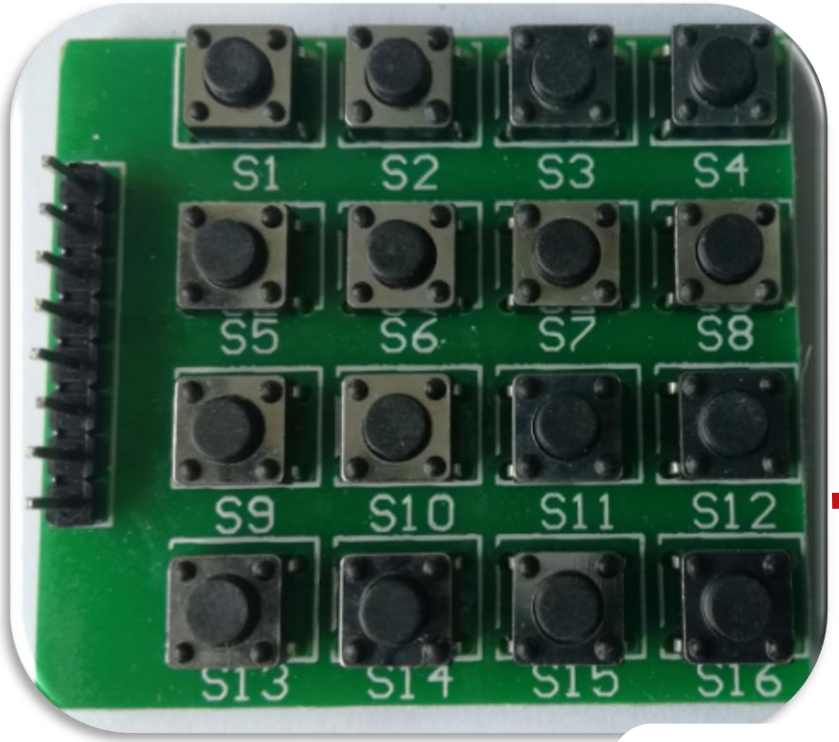
管脚分布



4个行管脚

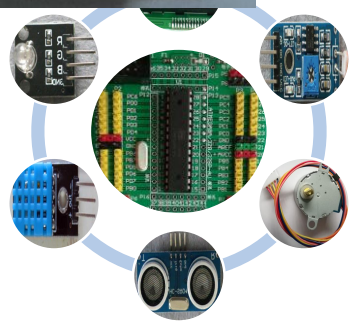
4个列管脚

4×4按键阵列的中断编程示例



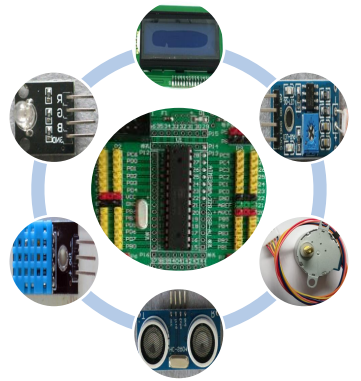
中断方式将小键盘最近四次按下的键，按顺序编码并显示到LCD

同时为了连线的方便，可改动接线顺序，编码顺序等



4×4按键阵列的中断编程示例代码1

```
#define与#include等省略
unsigned char hexStr[17]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','-'};
unsigned char key_no[5]={16,16,16,16,0};//存放4位连续按下的编码,以’ \0'结束
unsigned char getkey,keyno;//取按键阵列扫描输入，按下按键的编码
int main(void) { DDRD = (0xf0);//PD7~4为输出S_R0~3，PD3~0为输入S_C0~3
   PORTD = (0xff);//PD7~4输出高电平,PD3~0的内部上拉电阻启用
   TCCR0 = (1<<CS02)|(1<<CS00);//1024分频
   TCNT0 = 102;//从102开始计数，1MHz/1024/154约为150ms中断一次
   TIMSK |=(1<<TOIE0);//允许TOV0中断
   TWI_Init(); LCD_Init();
   sei();//全局中断开
   while (1) { LCD_Write_Char(0,1,hexStr[key_no[3]]);
               LCD_Write_NewChar(hexStr[key_no[2]]);
               LCD_Write_NewChar(hexStr[key_no[1]]);
               LCD_Write_NewChar(hexStr[key_no[0]]);
                 _delay_ms(20); } }
```

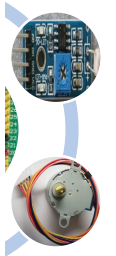


4×4按键阵列的中断编程示例代码2

```
ISR(TIMER0_OVF_vect)//溢出中断TCNT0=255溢出到102
{ TCNT0 = 102;//从102开始计数，1MHz/1024/154约为150ms中断一次
  keyno = 16;//默认无按键按下
  //1.扫描第1行
  PORTD = ~(1<<PORTD7);//PORTD7为0,其它为1，送给第1行
  _delay_us(2);
  getkey = (PIND & 0x0f);//取按键阵列的列状态
  switch(getkey) {     case 0x07:keyno = 0;break;//1行1列编码为0/1 //S4
         case 0x0b:keyno = 1;break;//1行2列编码为1/2 //S8
         case 0x0d:keyno = 2;break;//1行3列编码为2/3 //S12
         case 0x0e:keyno = 3;break;//1行4列编码为3/A //S16 }
  //2.扫描第2行
  PORTD = ~(1<<PORTD6);//PORTD6为0,其它为1，送给第2行
  _delay_us(2);
  getkey = (PIND & 0x0f);//取按键阵列的列状态
  。 。 。 。 。 。
```

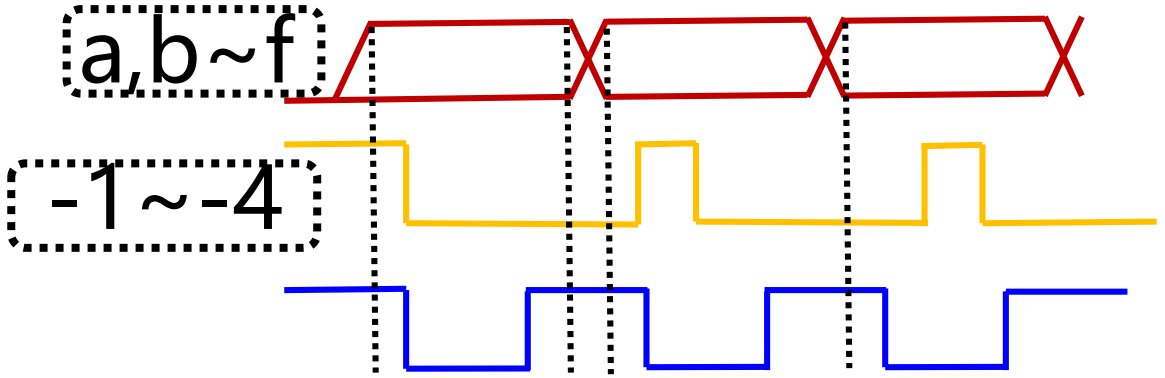
4×4按键阵列的中断编程示例代码3

```
。。。。。。
  //3.扫描第3行
。。。。。。
  //4.扫描第4行
  PORTD = ~(1<<PORTD4);//PORTD4为0，其它为1，送给第4行
  _delay_us(2); getkey = (PIND & 0x0f);//取按键阵列的列状态
  switch(getkey) {  case 0x07:keyno =12;break;//4行1列编码为12/*
          case 0x0b:keyno =13;break;//4行2列编码为13/0
          case 0x0d:keyno =14;break;//4行3列编码为14/#
          case 0x0e:keyno =15;break;//4行4列编码为15/D }
  /*一轮按键阵列处理结束*/
  if(keyno<16) { key_no[3]=key_no[2]; //移动按键数据
                 key_no[2]=key_no[1];
                 key_no[1]=key_no[0];
                 key_no[0]=keyno; //新的按键数据 }
} //ISR结束
```

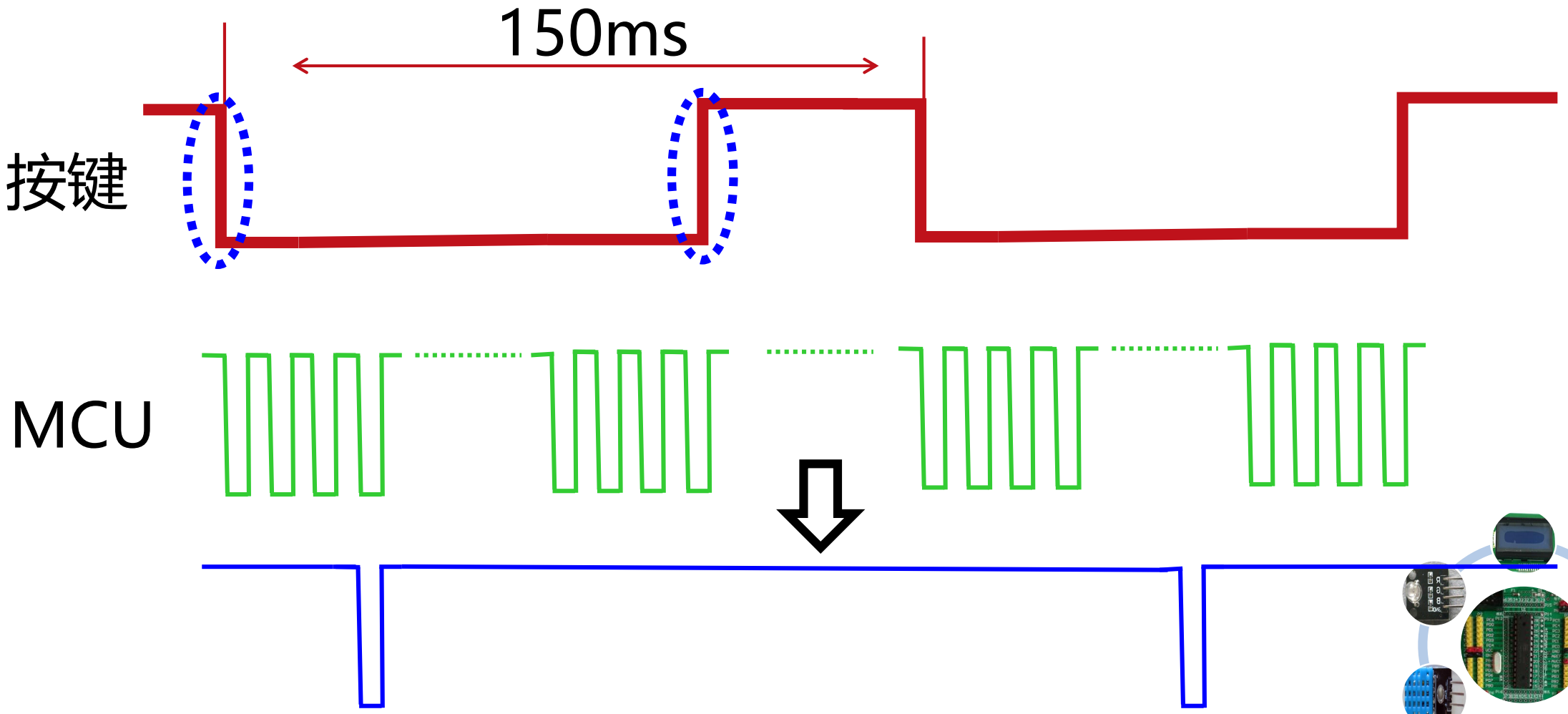


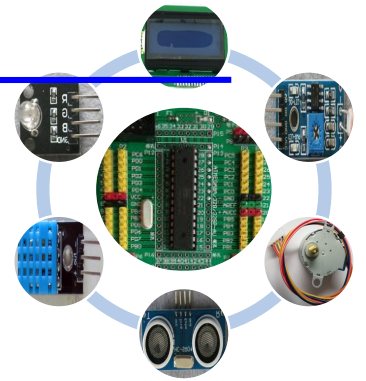
按键阵列与共阴极七段数码管知识拓展

- 七段数码管动态扫描显示会出现叠加现象
  - 避免数码管选中时数据抖动
- 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
  - 降低处理时间：统计、延时
  - 改进处理方式：边沿、中断

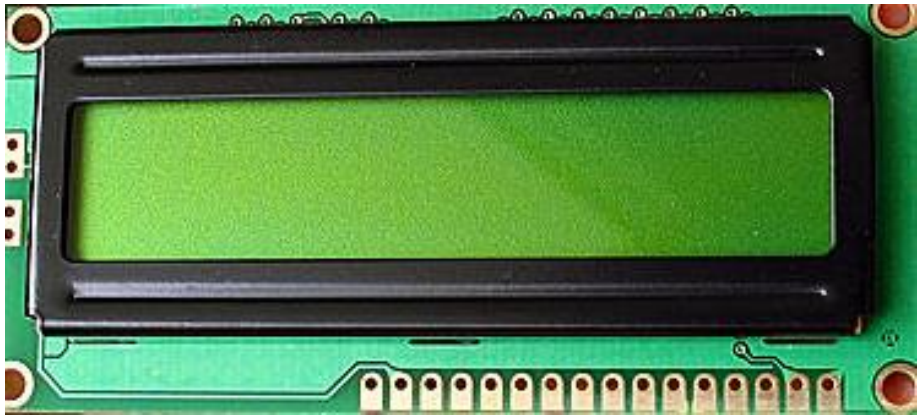


按键阵列与七段数码管知识拓展

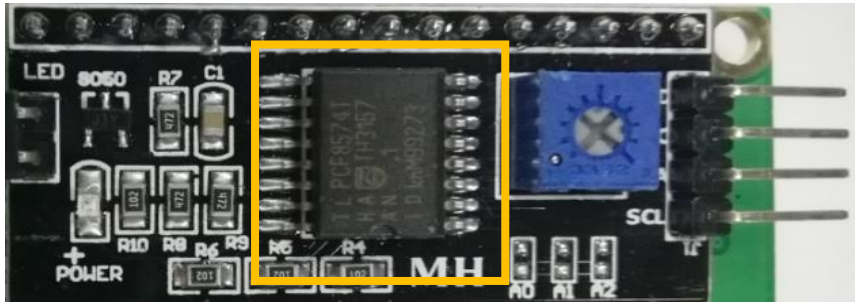




LCD1602接口的改进：减少接口连线

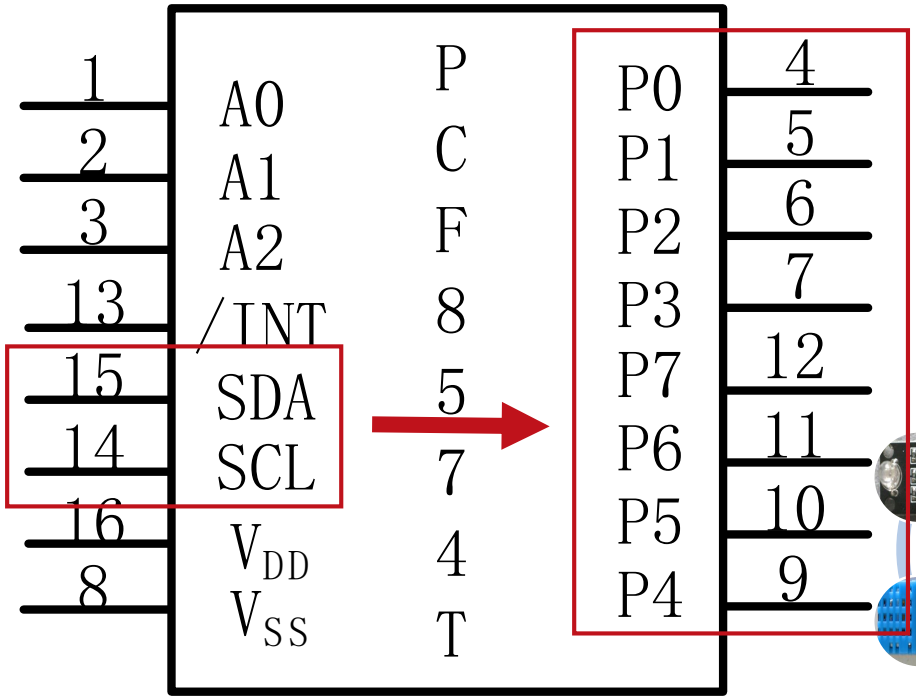


2bits-8bits

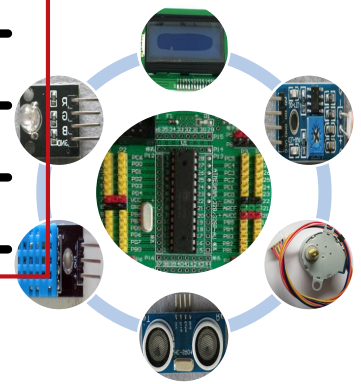


- 需要11/7Pins

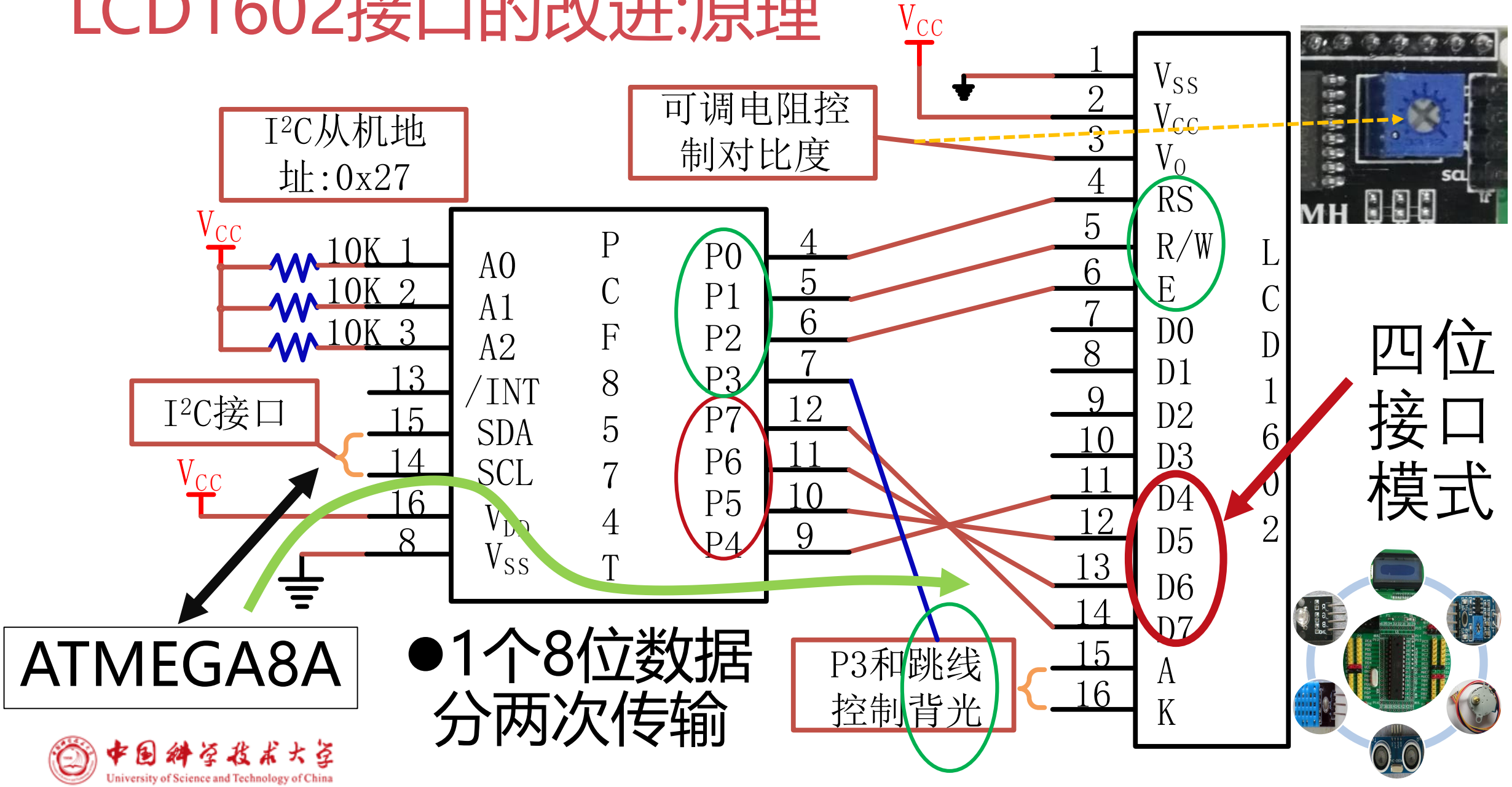
支持8位或4位数据传输模式



■MCU与LCD间仅需2Pins



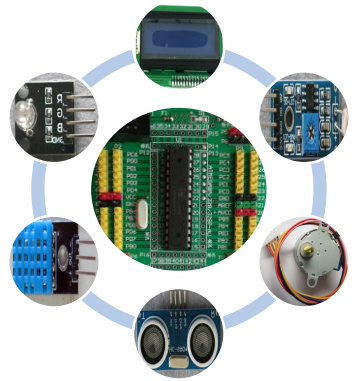
LCD1602接口的改进:原理



LCD1602液晶屏的显示控制

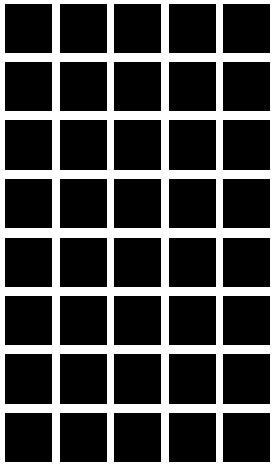
		1	2	3	4	5	6	…	40
DDRAM地址	行1	00h	01h	02h	03h	04h	05h	…	27h
	行2	40h	41h	42h	43h	44h	45h	…	67h

- 控制器型号为HD44780
- 3种存储空间：DDRAM、CGROM和CGRAM
  - DDRAM(Display Data RAM) 存储要显示的字符 ASCII码
  - CGROM(Character Generator ROM)为点阵数据（192）
  - **CGRAM存储用户自定义的字符点阵数据(最多8个)**



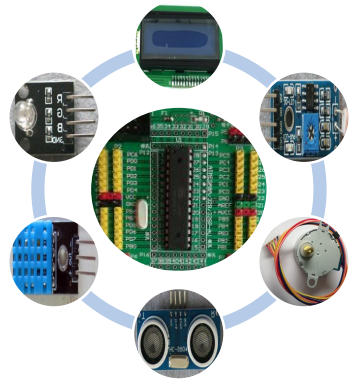
CGROM

前8个空间是自定义的CGRAM：8个5×8字符或4个5×10字符，其它为CGROM(固定字符的点阵数据)



b7-b4 / b3-b0	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)		0	@	P	`	p		ー	タ	ミ	α	p
0001	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ä	q
0010	(3)	"	2	B	R	b	r	「	イ	ツ	メ	ß	θ
0011	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
0100	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
0101	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
0110	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
0111	CG RAM (8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	q	π
1000	CG RAM (1)	(	8	H	X	h	x	ィ	ク	ネ	リ	√	x
1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
1010	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
1011	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	㎏	万
1100	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
1101	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
1110	(7)	.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
1111	CG RAM (8)	/	?	O	_	o	←	ッ	ソ	マ	゜	ö	

此编码用于兼容显示时的ASCII码，实际上每个字符占用5*8或5*10位



LCD1602的指令

指令名称	指令码										说明	时间@270KHz
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
清除屏幕	0	0	0	0	0	0	0	0	0	1	清屏,AC=0,光标复位	1.64ms
光标回原点	0	0	0	0	0	0	0	0	1	x	光标回原点，显示不变	1.64ms
进入模式设置	0	0	0	0	0	0	0	1	I/D	S	设置光标移动方向等	40us
屏幕开/关控制	0	0	0	0	0	0	1	D	C	B	屏幕,光标等显示切换	40us
光标或显示移位	0	0	0	0	0	1	S/C	R/L	x	x	光标和显示移位控制等	40us
功能设置	0	0	0	0	1	DL	N	F	x	x	8/4位接口,2/1行,5×8/10	40us
置CGRAM地址	0	0	0	1	ACG[5:0]						设置CGRAM地址到AC	40us
置DDRAM地址	0	0	1	ADD[6:0]							设置DDRAM地址到AC	40us
读忙标志和AD	0	1	BF	AC[6:0]							不论内部是否工作,均可读	40us
往RAM写数据	1	0	Write Data[7:0]								往CG/DDRAM写数据	40us
从RAM读数据	1	1	Read Data[7:0]								从CG/DDRAM读数据	40us
注解 ‘x’表示不用在意是‘0’还是‘1‘。	I/D=1:递增模式,I/D=0:递减模式; S=1:移位; S/C=1:显示移位,S/C=0:光标移位; R/L=1:右移,R/L=0:左移; DL=1:8位通信接口,DL=0:4位的; N=1:2行,N=0:1行; F=1:5×10点阵,F=0: 5×8点阵; BF=1:执行内部功能,BF=0:接收命令										AD:Address DDRAM:Display Data RAM CGRAM:Character Generator RAM ACG:CGRAM AD ADD:DDRAM&Cursor AD AC:Address counter for DRAM/CGRAM	

LCD1602指令与I²C接口的映射（4位模式）

■清除屏幕指令

RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1

I²C	D7	D6	D5	D4	AK	E	RW	RS
	0	0	0	0	1	1	0	0
	0	0	0	1	1	1	0	0

第1次发

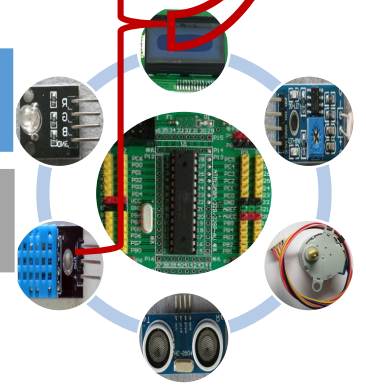
第2次发

■设置DDRAM地址指令

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	A6	A5	A4	A3	A2	A1	A0

■向DDRAM写数据指令

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	0	8位数据							



LCD命令示例代码（1）

```
unsigned char sf_data[8][8]={//自定义字符操作:最多8个自定义字符，也可以
汉字 {0x04,0x04,0x1F,0x15,0x15,0x1F,0x04,0x04}, //中
{0x1F,0x11,0x1F,0x15,0x1F,0x17,0x1F,0x1F}, //国
{0x0A,0x1E,0x0E,0x1A,0x0F,0x1A,0x0A,0x0A}, //科
{0x00,0x04,0x04,0x1F,0x04,0x00,0x0A,0x11}, //大
{0x00,0x02,0x1F,0x15,0x0A,0x12,0x05,0x09}, //欢
{0x01,0x00,0x07,0x01,0x02,0x02,0x02,0x01}, //迎
{0x04,0x08,0x17,0x15,0x1F,0x15,0x04,0x1F}, //迎
{0x00,0x0A,0x17,0x01,0x12,0x17,0x13,0x16}  //你 };
void LCD_CGR_Write()//将自定义的8个字符数据写入CGRAM
{ unsigned char i,j;
  for(i=0;i<8;i++) for(j=0;j<8;j++)
  { LCD_8Bit_Write(0x40+i*8+j,0);//确定地址
    LCD_8Bit_Write(sf_data[i][j],1);//将对应的自定义字符数据写入
  }
}
```

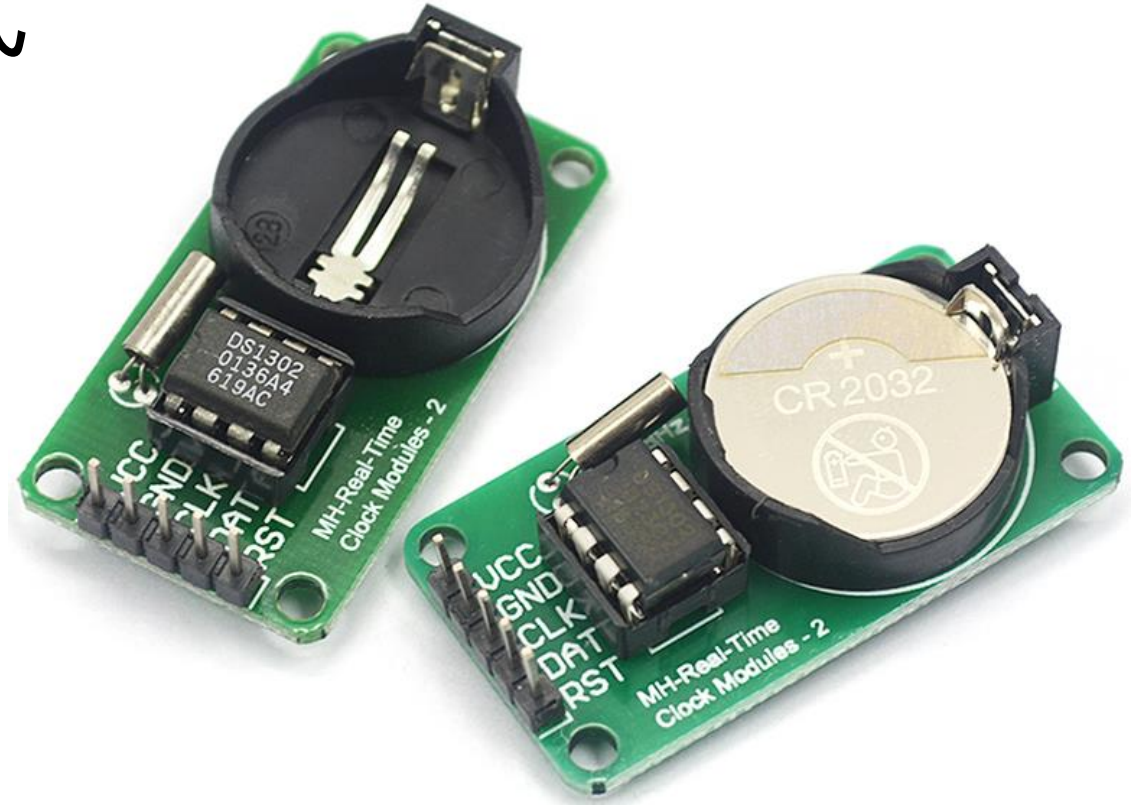
LCD命令示例代码（2）

```
int main(void) { int i;
  TWI_Init(); LCD_Init(); LCD_CGR_Write(); //初始化TWI和LCD，写入8个自定义
  LCD_Write_String(0,0,"LCD1602 Command 20230425"); _delay_ms(2000);
  LCD_8Bit_Write(0x01,0);//清屏 _delay_ms(2);
  LCD_Write_String(0,0,"LCD1602 Command 20230425");
  LCD_8Bit_Write(0x18,0);_delay_ms(1); //左移一个字符
  LCD_8Bit_Write(0x18,0);_delay_ms(1);
  LCD_8Bit_Write(0x18,0);_delay_ms(1);
  LCD_8Bit_Write(0x18,0);_delay_ms(1);
  LCD_8Bit_Write(0x18,0);_delay_ms(1);
  _delay_ms(2000);
  for(i=0;i<8;i++)LCD_Write_Char(1,i,i); //显示自定义字符
  _delay_ms(2000);
  LCD_8Bit_Write(0x1C,0);_delay_ms(1); //右移一个字符
  LCD_8Bit_Write(0x1C,0);_delay_ms(1);
  LCD_8Bit_Write(0x1C,0);_delay_ms(1);
  LCD_8Bit_Write(0x1C,0);_delay_ms(1);
  LCD_8Bit_Write(0x1C,0);_delay_ms(1);
 while (1) { //LCD_8Bit_Write(0x18,0);_delay_ms(500); } }
```

实时时钟模块：DS1302带电池

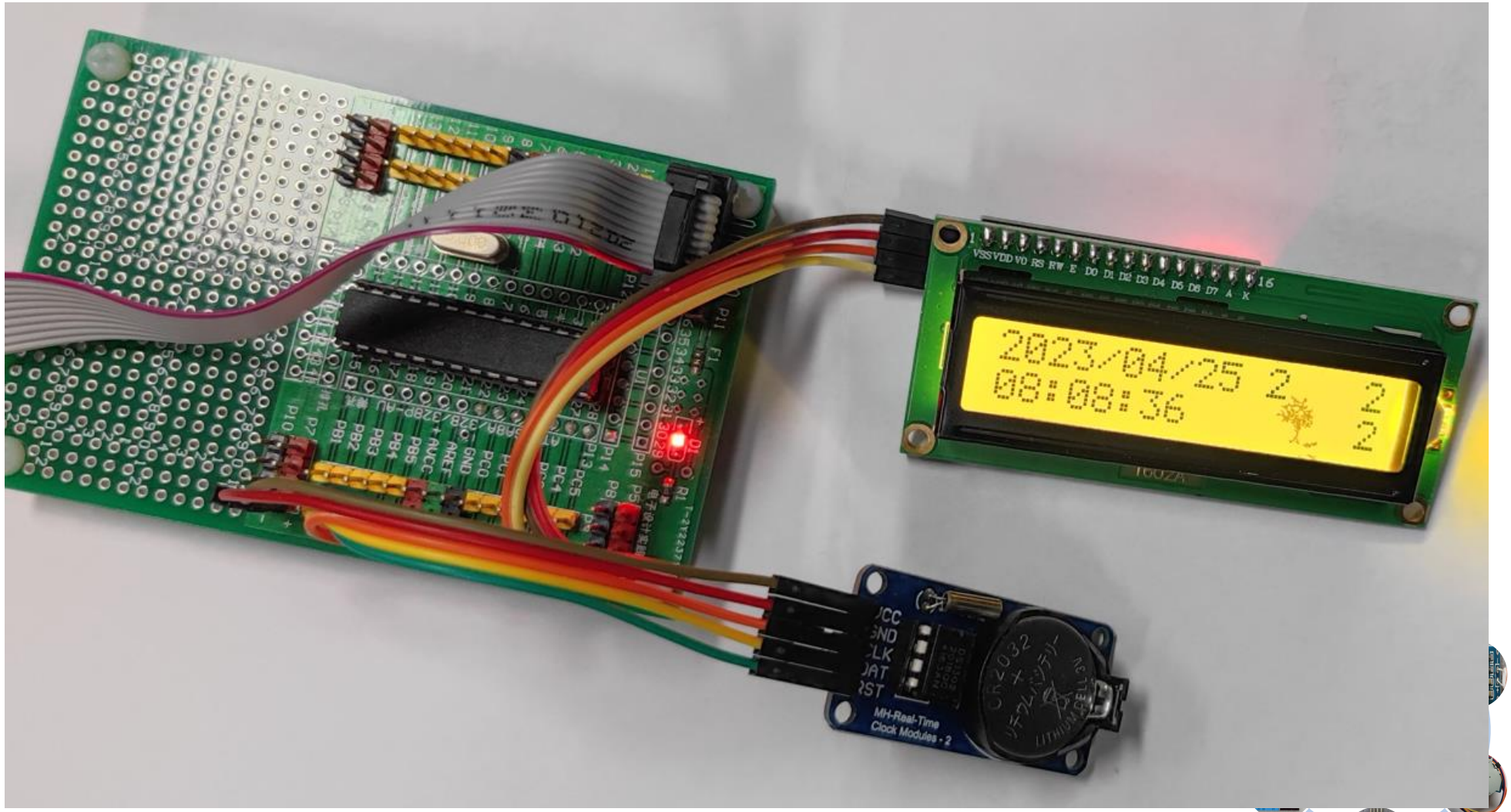
- 内含实时时钟/日历和31 个字节的静态RAM
- 提供秒、分、时、日、周、月、年的信息，每月的天数和闰年的天数可自动调整。时钟操作可通过AM/PM 指示决定采用24 或12 小时格式

- DS1302 与单片机之间能简单地采用同步串行的方式进行通信，仅需三个IO连线：（1）RST 复位（2）I/O 数据线（3）SCLK串行时钟



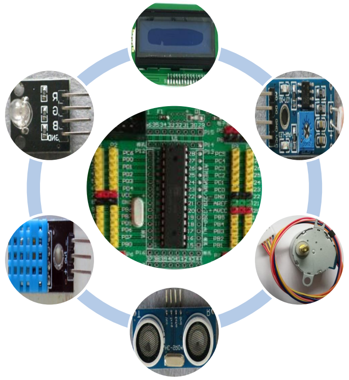
DS1302实时时钟模块示例

- 在LCD1602显示日期和时间



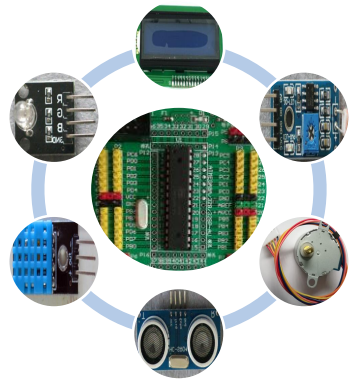
DS1302实时时钟模块示例头文件代码（1）

```c
。。。。。。。。。。。。。。。
/*DS1302与ATmega8a的接口  PC2~SCLK,PC1~I/O,PC0~/RST  */
//DS1302地址定义

。。。。。。。。。。。。。。。
//初始时间定义
unsigned char  time_buf[8] = {0x20,0x23,0x04,0x25,0x08,0x08,0x08,0x02};//初始时间2023年4月25号8点8分8秒 星期二
unsigned char dis_time_buf[16]={0};//存放要显示的时间日期数据
//DS1302初始化函数
void ds1302_init(void)/*DS1302与MCU接口 PC2~SCLK,PC1~I/O,PC0~/RST */
{
  DDRC|=(1<<DDRC2)|(1<<DDRC1)|(1<<DDRC0);//PC2,1,0为输出
  //DDRC &=~(1<<DDRC1);//PC1(I/O:DATA)暂为输入
  PORTC &= ~(1<<PORTC0);//RST脚置低
  PORTC &= ~(1<<PORTC2);//SCK脚置低
}
```



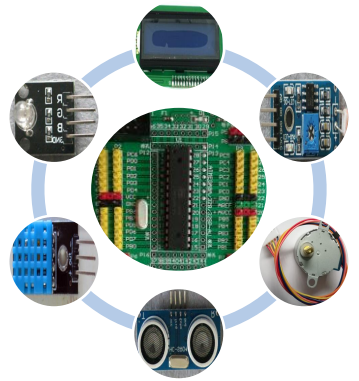
DS1302实时时钟模块示例头文件代码（2）

```
void ds1302 write_byte(unsigned char addr, unsigned char d) //向DS1302写入一字节数据
{  unsigned char i;
  PORTC |=(1<<PORTC0);//RST=1,启动DS1302总线
  //写入目标地址：addr
  addr = addr & 0xFE;  //最低位置零，寄存器0位为0时写，为1时读
  for (i = 0; i < 8; i ++) {//1Byte=8bit，从位0开始发送
    if (addr & 0x01)  PORTC |=(1<<PORTC1);//PC1(IO)输出1
    else PORTC &=~(1<<PORTC1);//PC1(IO)输出0
      delay us(1);
   PORTC |=(1<<PORTC2);////产生时钟：PC2(SCLK)输出1
      delay us(2);
    PORTC &=~(1<<PORTC2);//PC2(SCLK)输出0
      delay us(2);
    addr = addr >> 1;
  } //end for loop
```



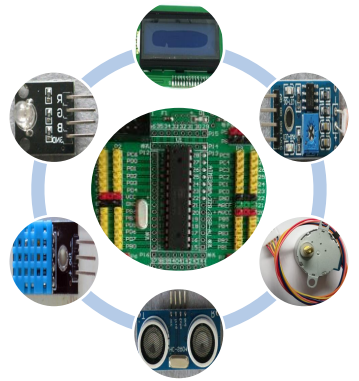
DS1302实时时钟模块示例头文件代码（3）

```
//写入数据：d
for (i = 0; i < 8; i ++) {//1Byte=8bit，从位0开始发送
  if (d & 0x01) {     PORTC |=(1<<PORTC1);//PC1(IO)输出1
  }
  else { PORTC &=~(1<<PORTC1);//PC1(IO)输出0
  }
    delay us(1);
  PORTC |=(1<<PORTC2);////产生时钟：PC2(SCLK)输出1
    delay us(2);
  PORTC &=~(1<<PORTC2);//PC2(SCLK)输出0
    delay us(2);
  d = d >> 1;
} //end for loop 2
 PORTC &=~(1<<PORTC0);//RST=0,停止DS1302总线
} //end for ds1302_write_byte function
```



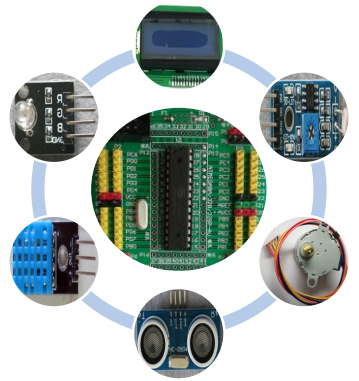
DS1302实时时钟模块示例头文件代码（4）

```
//从DS1302读出一字节数据
unsigned char ds1302_read_byte(unsigned char addr) {
  unsigned char i,temp=0;
  PORTC |=(1<<PORTC0);//RST=1,启动DS1302总线
  //写入目标地址：addr
  addr = addr | 0x01;  //最低位置1，寄存器0位为0时写，为1时读
  for (i = 0; i < 8; i ++) {//1Byte=8bit，从位0开始发送
    if (addr & 0x01) {PORTC |=(1<<PORTC1);//PC1(IO)输出1 }
    else { PORTC &=~(1<<PORTC1);//PC1(IO)输出0 }
     delay_us(1);
    PORTC |=(1<<PORTC2);////产生时钟：PC2(SCLK)输出1
     delay_us(2);
    PORTC &=~(1<<PORTC2);//PC2(SCLK)输出0
     delay_us(2);
    addr = addr >> 1;
  } //end for loop 1
```



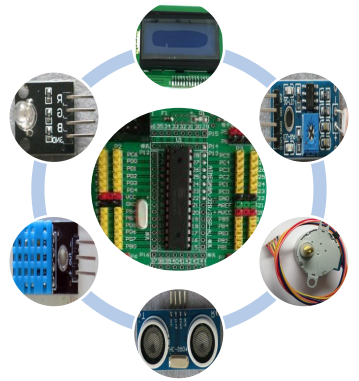
DS1302实时时钟模块示例头文件代码（5）

```
//读取数据到temp
DDRC &=~(1<<DDRC1);//PC1(IO),为输入
for (i = 0; i < 8; i ++) {//从0位开始接收
  temp = temp >> 1;//每接收1位放在最高位
  if (PINC & (1<<PINC1)) {//PC1管脚为1或0
    temp |= 0x80;//收到1     }
  else { temp &= 0x7F;//收到0 }
   delay us(1);
  PORTC |=(1<<PORTC2);////产生时钟：PC2(SCLK)输出1
   delay us(2);
  PORTC &=~(1<<PORTC2);//PC2(SCLK)输出0
   delay us(2);     }
DDRC |=(1<<DDRC1);//恢复PC1(IO),为输出
PORTC &=~(1<<PORTC0);//RST=0,停止DS1302总线
return temp;
}。。。。。。。。。。。。。。。。
```



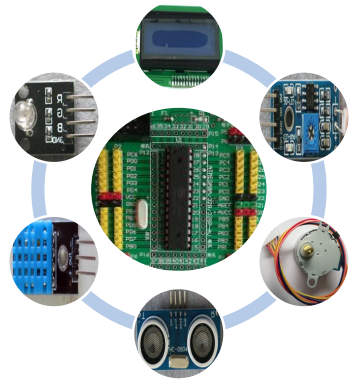
DS1302实时时钟模块示例main代码（1）

```
#define F_CPU 8000000UL  //ATmega8a工作时钟
#include <avr/io.h>
#include "twi_lcd.h"
#include "DS1302.h"
unsigned char Read_E2PROM(unsigned int uiAddr)
{ while(EECR & (1<<EEWE));  /*等待上次写结束*/
  EEAR = uiAddr;  /*设置读取EEPROM的地址*/
  EECR |= (1<<EERE);  /*通过EERE位读取EEPROM,*/
  return EEDR; /*返回读取的数据*/
}
void Write_E2PROM(unsigned int uiAddr,unsigned char ucData)
{ while(EECR & (1<<EEWE));  /*等待上次写结束*/
  EEAR = uiAddr;  /*设置要写EEPROM的地址*/
  EEDR = ucData;  /*要写的数据存入寄存器*/
  EECR |= (1<<EEMWE);  /*EEMWE置为‘1’，*/
  EECR |= (1<<EEWE);   /*通过EEWE位写EEPROM，*/
}
```



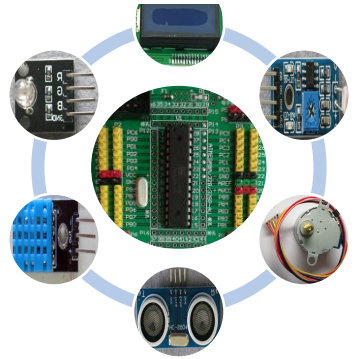
DS1302实时时钟模块示例main代码（2）

```
int main(void)
{
  unsigned char e2d=0,r2d=0;
  ds1302_init();//初始化ds1302

   delay_ms(50);//等待系统稳定
  TWI_Init();//初始化TWI接口
  LCD_Init();//初始化LCD1602
  _delay_ms(10);//
```



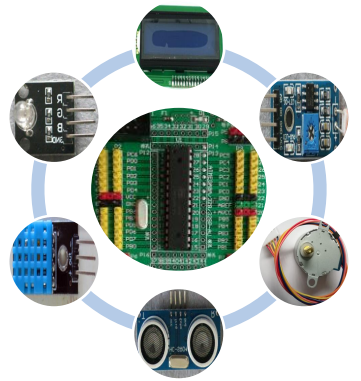
DS1302实时时钟模块示例main代码（3）

```
r2d=ds1302 read byte(0xC0);//读取ds1302 Ram的0x00地址数据
if(r2d!=1)//初始化ds1302,根据ds1302 RAM 0x00地址的数据，仅初始化一次
{ ds1302 write time();//写入指定的日期和时间到ds1302
   delay us(10);
  ds1302 write byte(ds1302 control add,0x00);//关闭写保护
  ds1302 write byte(0xC0,1);//RAM空间0x00地址写入1
  ds1302 write byte(0xC2,1);//初始化RAM空间0x01地址为1，记录断电和程序
下载次数，测试ds1302断电后数据不丢失
  ds1302_write_byte(ds1302_control_add,0x80);//打开写保护

  Write E2PROM(0x001,1);//初始化EEPROM空间0x01地址为1，记录断电和程
序下载次数，测试断电次数
  _delay_us(10);
}
```



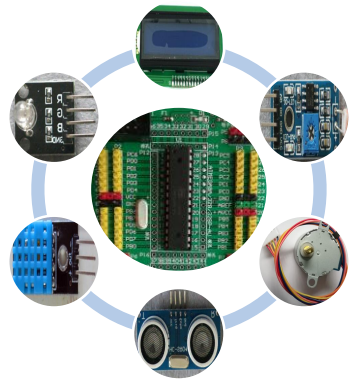
DS1302实时时钟模块示例main代码（4）

```
//用ds1302RAM中的0x01空间记录断电和程序下载的次数
r2d=ds1302 read_byte(0xC2);//读取ds1302 Ram的0x01地址数据
r2d++;//多一次
ds1302 write byte(ds1302 control add,0x00);//关闭写保护
ds1302 write byte(0xC2,r2d);//RAM空间0x01地址加1，记录断电和程序下载次数，测试ds1302断电后数据不丢失
ds1302_write_byte(ds1302_control_add,0x80);//打开写保护
//显示
for(e2d=15;e2d>11;e2d--)//LCD1602第一行显示RAM中记录的断电次数
{
  if(r2d==0)break;
  LCD Write_Char(0,e2d,r2d%10+'0'  );
  r2d /=10;
}
```



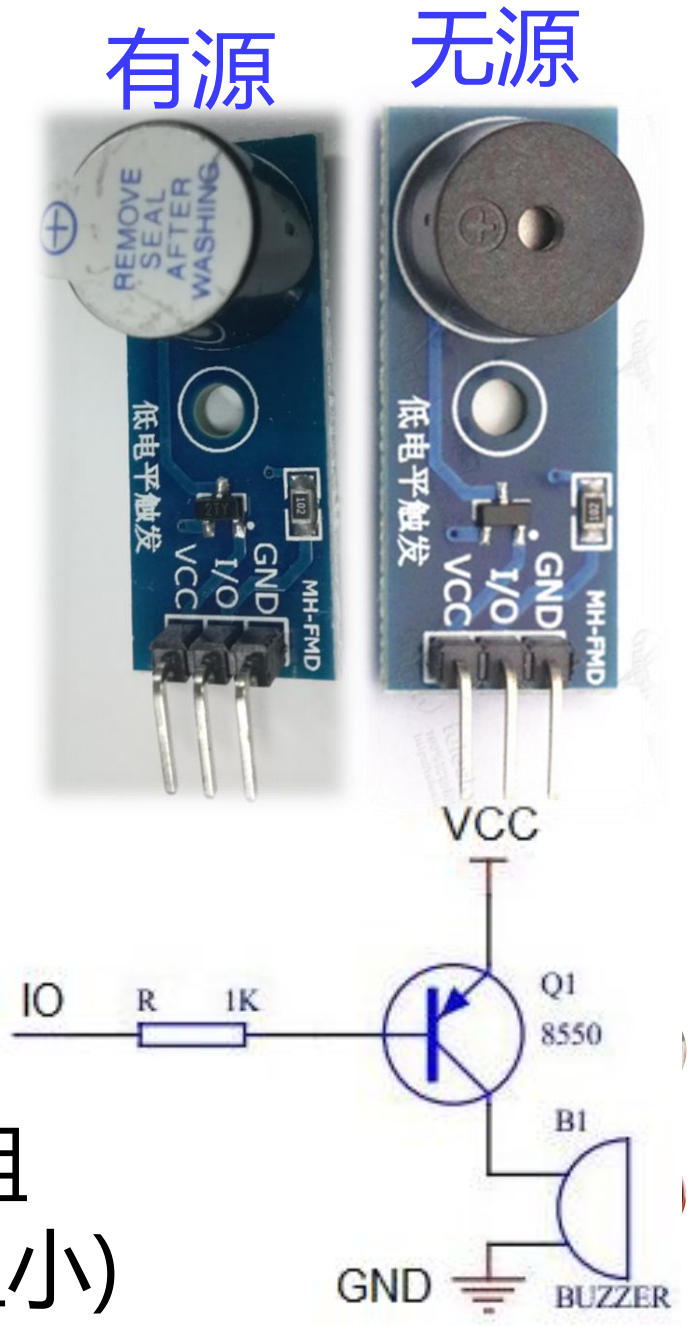
DS1302实时时钟模块示例main代码（5）

```
//用ATmega8a EEPROM记录断电和程序下载的次数
e2d=Read  E2PROM(0x001);//读取EEPROM的0x01地址数据
 delay  ms(1);   e2d++;//多一次
Write  E2PROM(0x001,e2d);//0x01地址内容加1，记录断电和程序下载次数
 delay  ms(1);
for(r2d=15;r2d>11;r2d--)//LCD1602第二行显示EEPROM中记录的断电次数
{ if(e2d==0)break;
  LCD  Write  Char(1,r2d,e2d%10+'0'   );
  e2d /=10;   }
 delay  ms(10);
while (1)
{ read  data();/*从ds1302中读取时间和日期数据*/     delay  ms(2);
  Display();/*在LCD1602显示日期和时间等*/    _delay_ms(200);
 }
}
```



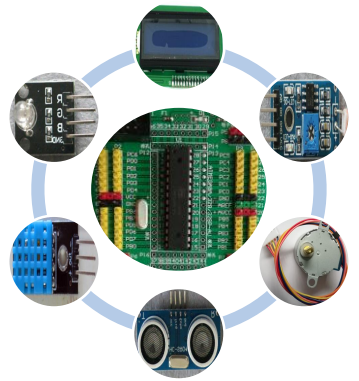
蜂鸣器：有源和无源

- 有源蜂鸣器：
  - 内部带震荡源，只要一通电就会叫
  - 程序控制方便，单片机一个低电平就可以让其发出声音
- 无源蜂鸣器：
  - 内部不带震荡源，用直流信号无法令其鸣叫。必须用方波去驱动它
  - 声音频率可控，可以做出“多来米发索拉西”的效果
- 如何判断：加电（有源持续响），测量电阻(无源电阻小)



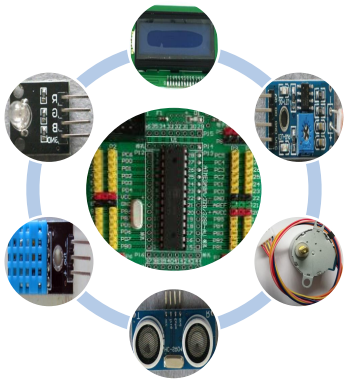
蜂鸣器：有源和无源示例代码（1）

```
#define  F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "twi_lcd.h"
unsigned char duty=50;//占空比，百分比
unsigned int freq=30;//30~8000Hz
ISR(INT0_vect)
{ if(duty > 90)//百分比    duty = 10;
  else    duty +=5;
}
ISR(INT1_vect)
{  if(freq >7999)  freq = 30;
   else  freq +=100;
}
```



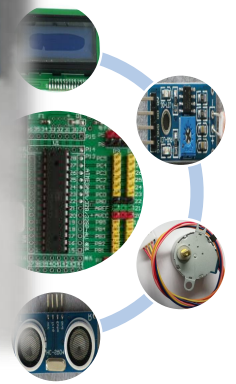
蜂鸣器：有源和无源示例代码（2）

```
int main(void)
{ TWI_Init();  LCD_Init();
  DDRD &=~((1<<DDRD2) |(1<<DDRD3));//INT0(PD2)和INT1(PD3)分别调整占空比和频率
  PORTD |=(1<<PORTD2)|(1<<PORTD3);//开启内部上拉电阻，即PD2和PD3管脚默认为高电平
  DDRB |=(1<<DDRB1);//PB1控制蜂鸣器的IO
  MCUCR |=(1<<ISC01)|(1<<ISC11);//INT0和INT1下降沿触发中断（执行对应的ISR）
  GICR |=(1<<INT0)|(1<<INT1);//开中断
  sei();//全局中断开
  unsigned int high,low,i;
  while (1)
  { high =  F_CPU/freq*duty/100;//
    low = F_CPU/freq - high;//
    if(low >1290)low-=1290;
    high /=12; low /=12;
    PORTB |=(1<<PORTB1);for(i=0;i<high;i++)  delay_us(1);
    PORTB &=~(1<<PORTB1);for(i=0;i<low;i++)_delay_us(1);
    }
}
```



霍尔模块

- 与光敏电阻模块类似
  - D0为数字开关输出（0或1）
  - A0为模拟电压输出
  - VCC为供电电源的正极
  - GND为供电电源的负极



综合实验要求

1，实验时间：第9、10周两周时间，最晚第11周星期一晚上（5月15日）实验时间验收，设计报告在12周周一之前交

2，综合设计要求：基于课程，独立完成一个电子系统设计，要完整，可以运行，其它不做要求

3，设计报告要求：设计思路、过程，必要的流程与代码分析，结果与分析，拓展等。代码要有必要的注释。最后把报告和代码等一起打包提交