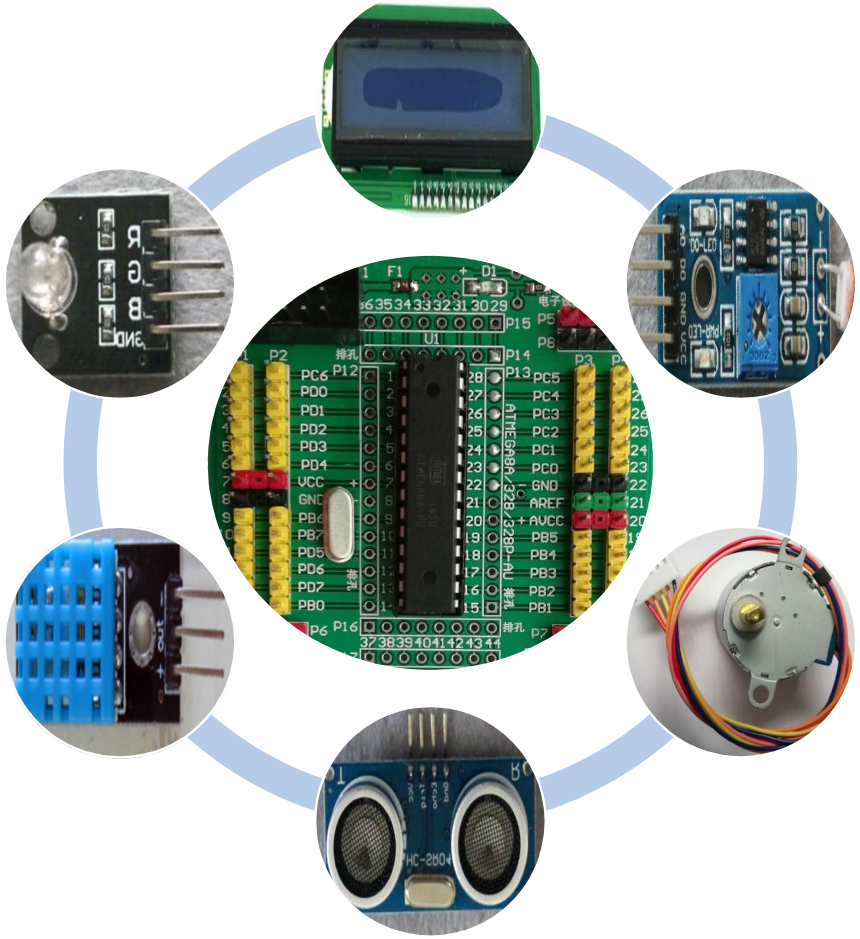
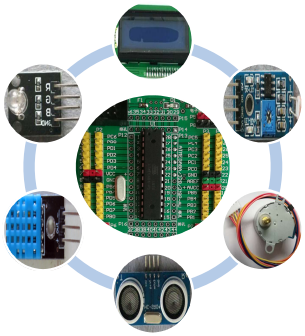
电子设计实践基础

MCU的TWI接口（I²C）与液晶屏（LCD1602）及其编程



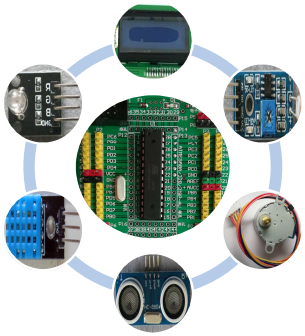
上节课内容回顾

- MCU**中断**、七段**数码管**、**按键阵列**及其编程
  - ATmega8A的**中断**及其程序设计
  - **共阴极**七段**数码管**及其编程
  - **按键**阵列及其编程



本节课主要内容

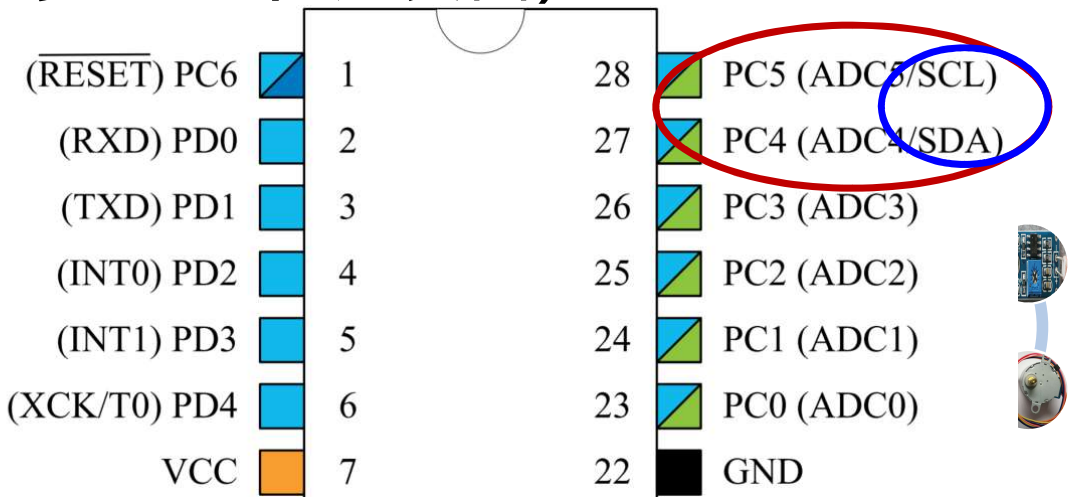
- MCU的TWI接口、LCD显示屏及其编程
  - **TWI**接口与编程
  - **LCD**1602模块与编程



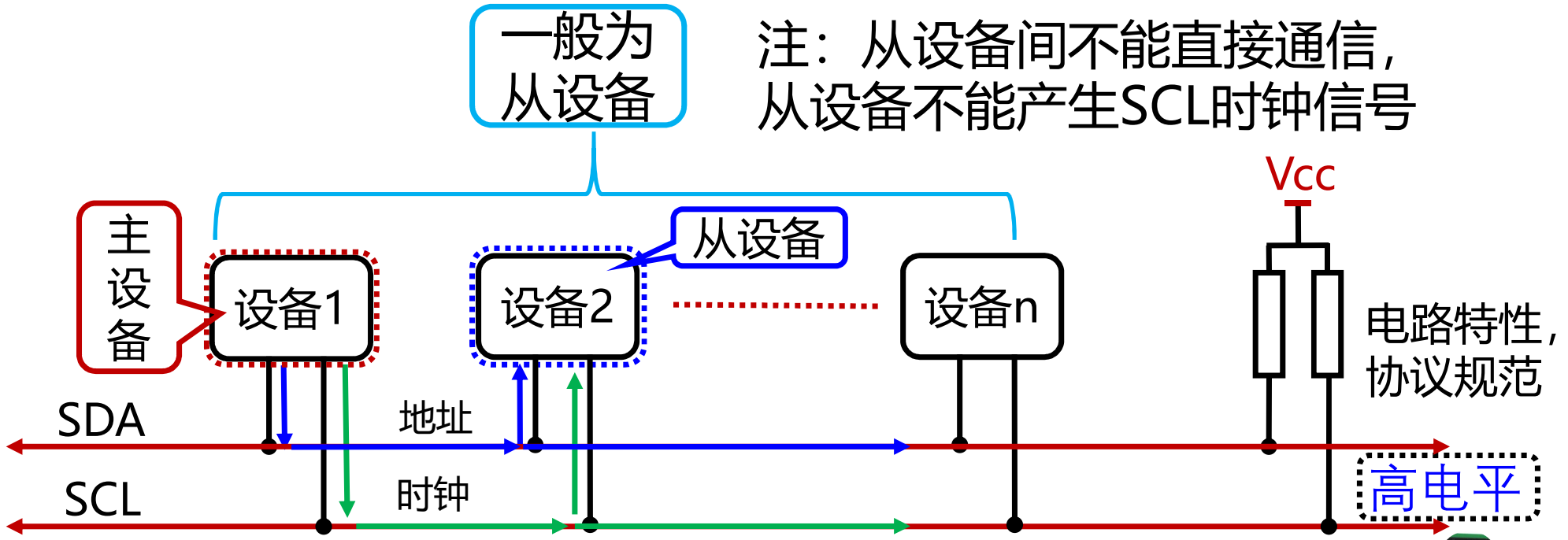
ATmega8A TWI接口

- TWI-Two Wire serial Interface：两线串行接口总线
  - SCL-Serial Clock：串行时钟线；
  - SDA-Serial Data：串行数据线
  - 与I^2C（Inter - Integrated Circuit）总线兼容
  - 最快400KHz的传输速度（*SCL频率，速率<400kbps*）
  - 支持主/从设备(7位地址:最多128个从设备)

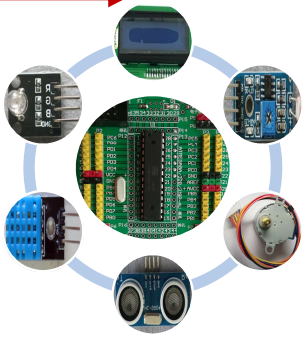
其实还有第三根线：GND，高低电平都是相对于参考地，不同供电的设备间需要共地才能通信



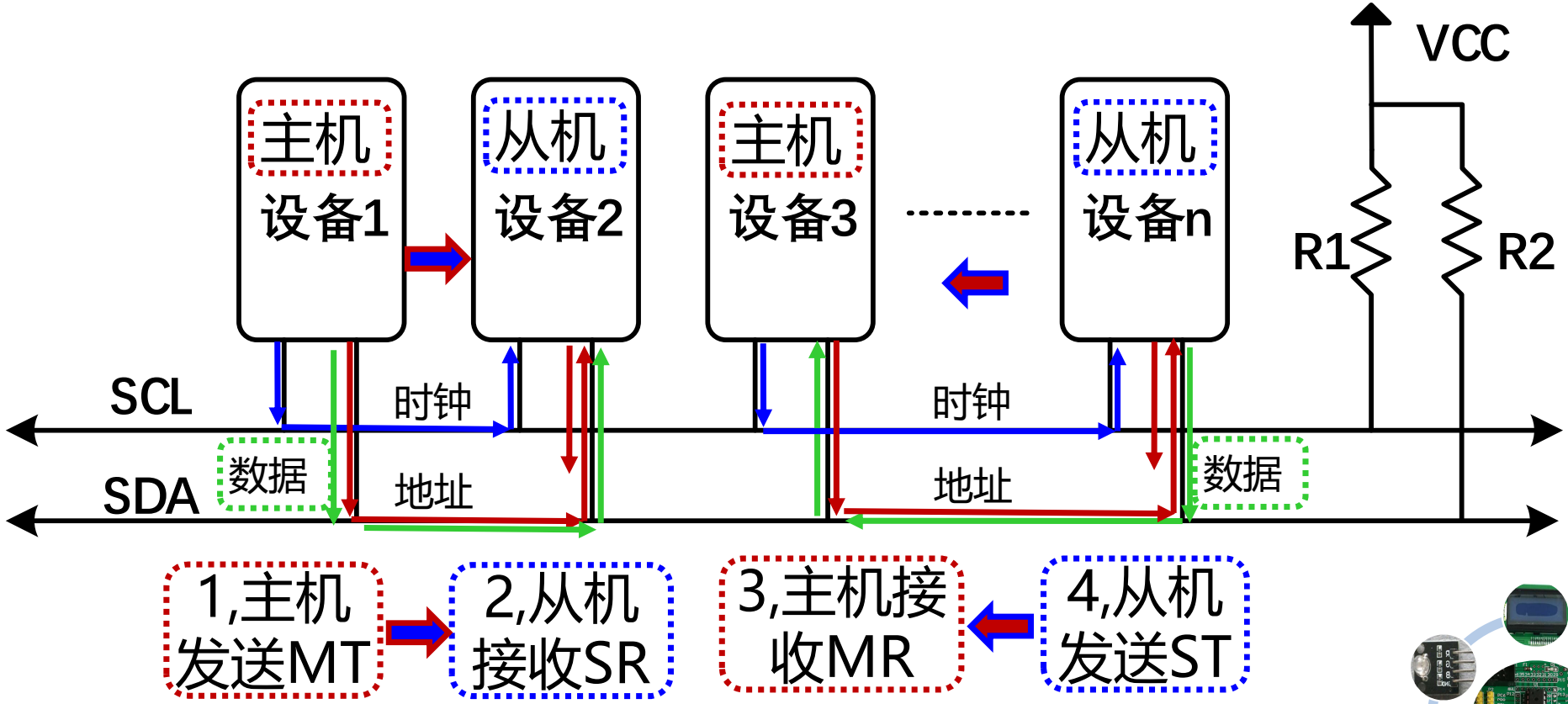
ATmega8A TWI接口的总线定义



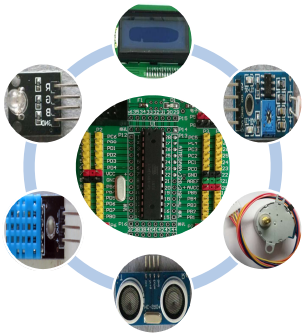
设备都有自己的地址，同时TWI协议还具有处理总线冲突的机制



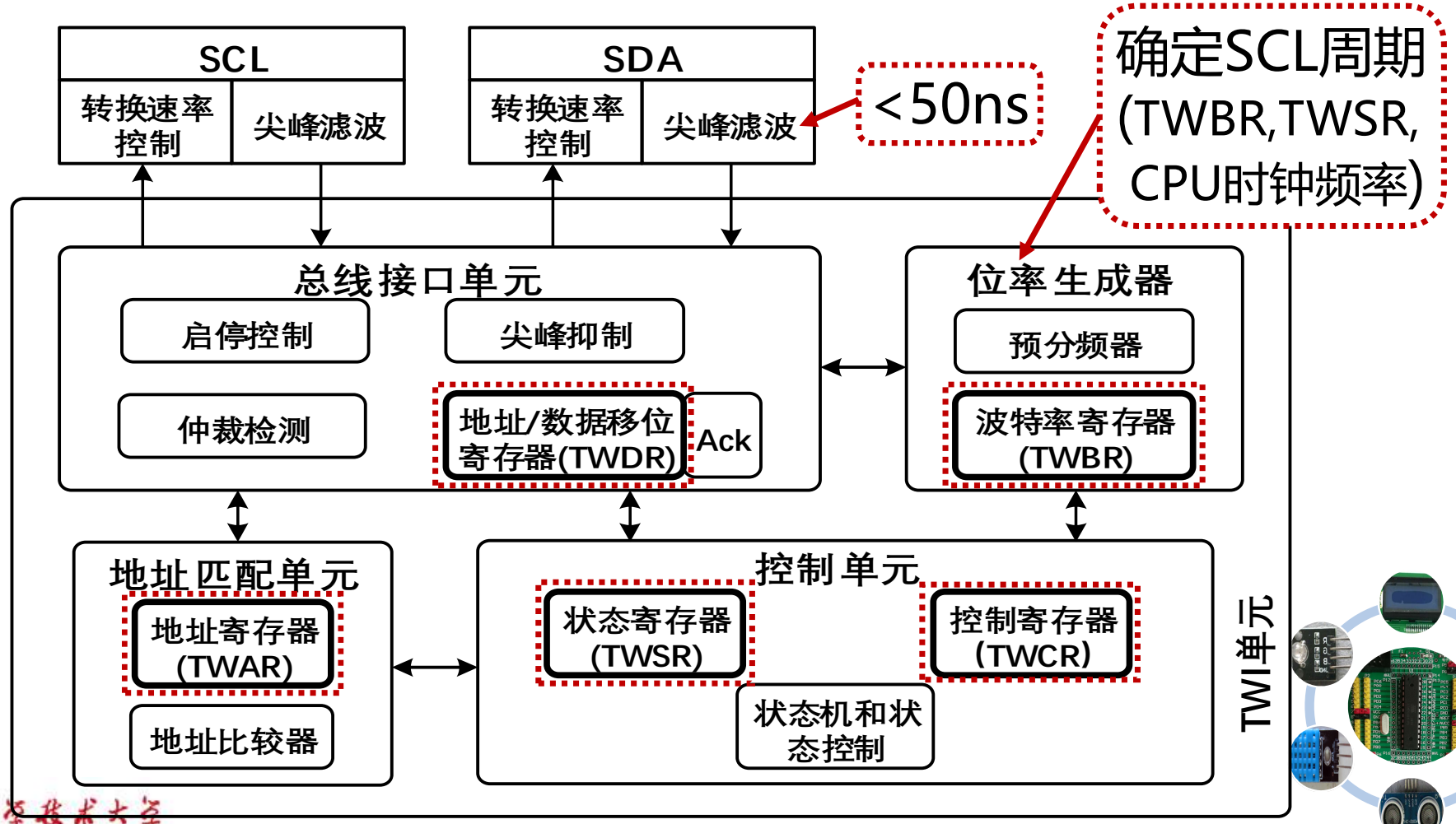
ATmega8A TWI接口的四种模式



M-Master,S-Slave,T-Transmitter,R-Receiver

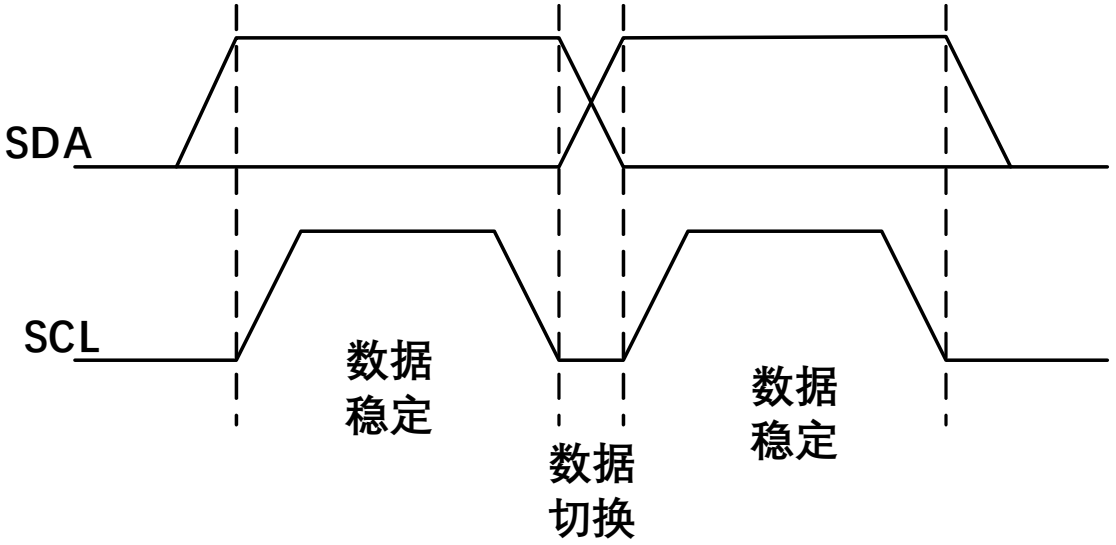


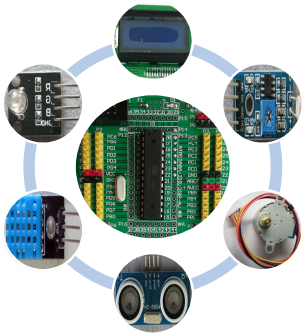
ATmega8A TWI接口的内部结构



ATmega8A TWI接口的位传输

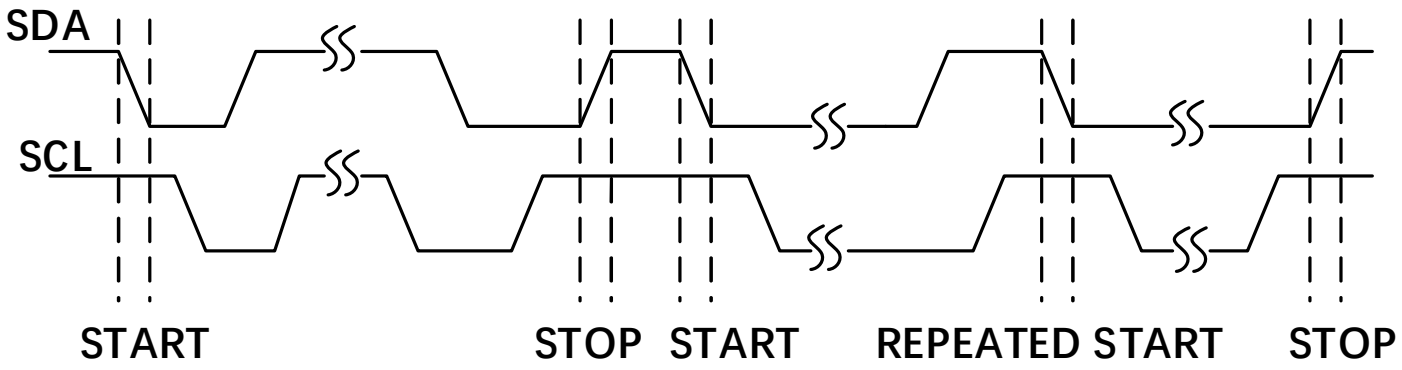
- SDA线上每个数据位的传输都与SCL线上的一个时钟脉冲同步
- SCL高电平时SDA线必须稳定

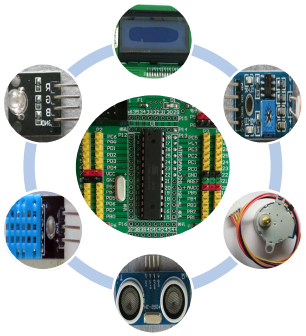




ATmega8A TWI接口的**启停**控制

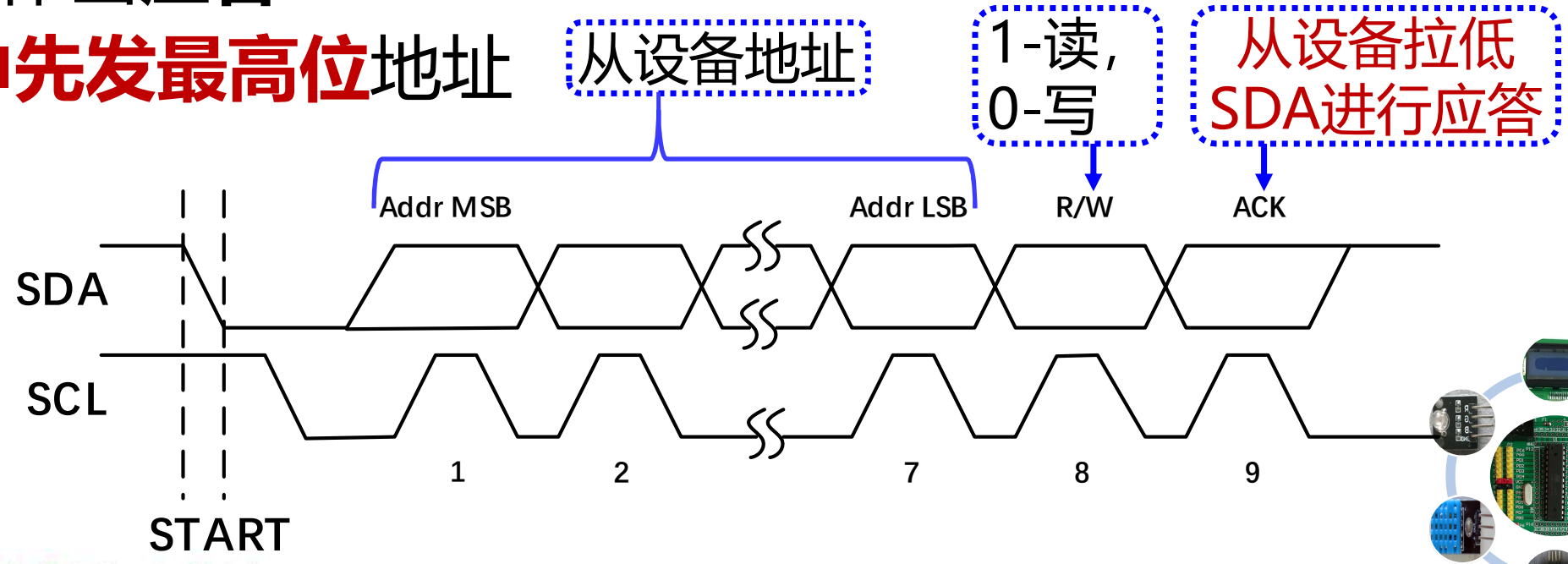
- 主设备在总线上发布START信号，**开启**传输；发布STOP信号，**结束**传输
- 在START和STOP之间可传输**1个字节**数据，此时总线忙，通过RESTART信号可连续传输**多字节**数据

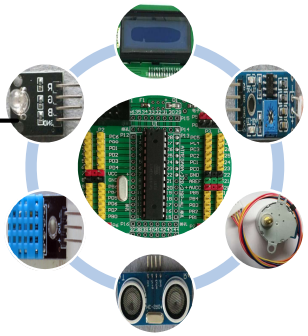




ATmega8A TWI接口的**地址包**格式

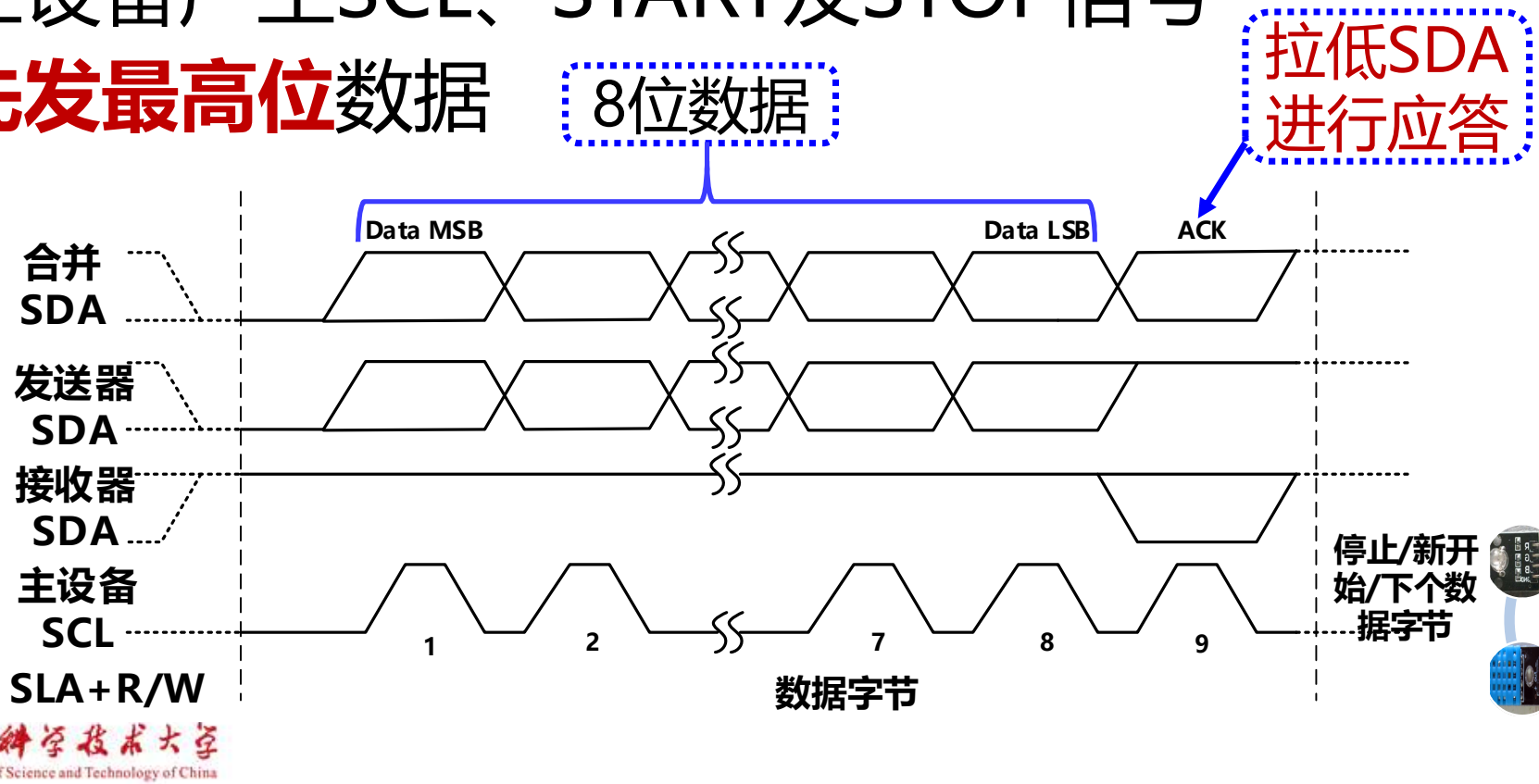
- 地址包长度为9位：7位地址，1位读写和1位应答
- 主设备寻址从设备，告知从设备是**读/写**，**从设备**作出**应答**
- **先发最高位**地址

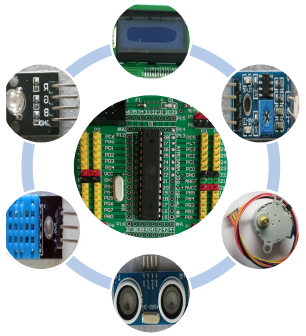




ATmega8A TWI接口的**数据包**格式

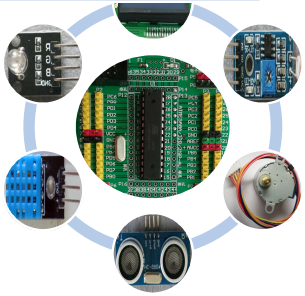
- 数据包的长度为**9位**：8位数据和1位应答
- 主设备产生SCL、START及STOP信号
- **先发最高位**数据





ATmega8A TWI接口的寄存器

寄存器名	偏移地址	说明
TWBR	0x00	TWI Bit Rate Register（位率）
TWCR	0x36	TWI Control Register（控制）
TWSR	0x01	TWI Status Register（状态）
TWDR	0x03	TWI Data Register（数据）
TWAR	0x02	TWI(slave) Address Register (地址)



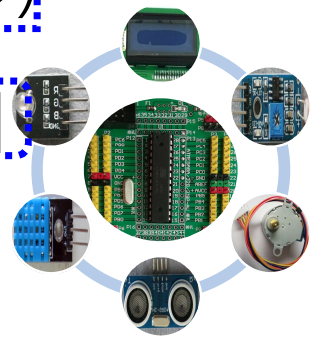
ATmega8A TWI位率寄存器-TWBR

Bit	7	6	5	4	3	2	1	0
TWBR	TWBR[7:0]							
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

$$f_{SCL} = \frac{CPU\ Clock\ frequency}{16 + 2 \times (\boldsymbol{TWBR}) \times (Prescaler\ Value)}$$

最大：1M/16=62.5K ← 1MHz

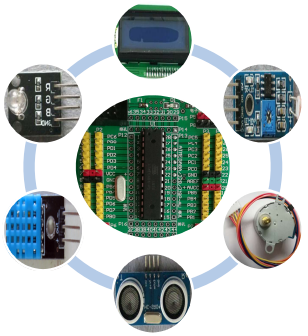
TWSR:TWPS[1:0]



ATmega8A TWI控制寄存器-TWCR

Bit	7	6	5	4	3	2	1	0
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

- TWINT：TWI中断标志位
- TWEA：TWI开启应答
- TWSTA：TWI发送START
- TWSTO：TWI发送STOP
- TWWC：TWI写冲突标志位
- TWEN：TWI开启/使能
- TWIE：TWI中断允许



ATmega8A TWI状态寄存器-TWSR

Bit	7	6	5	4	3	2	1	0
TWSR	TWS4	TWS3	TWS2	TWS1	TWS0		TWPS1	TWPS0
Access	R	R	R	R	R		R/W	R/W
Reset	0	0	0	0	1		0	0

TWI状态位

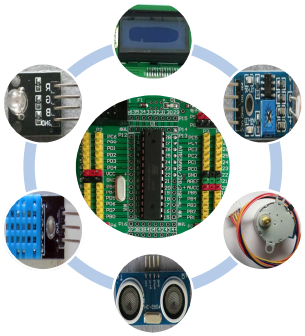
TWPS1	TWPS0	预分频值
0	0	1
0	1	4
1	0	16
1	1	64

$$f_{SCL} = \frac{CPU\ Clock\ frequency}{16 + 2 \times (TWBR) \times (Prescaler\ Value)}$$

ATmega8A TWI数据寄存器-TWDR

Bit	7	6	5	4	3	2	1	0
TWDR	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0
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

通过TWI接口发送与接收数据均需用TWDR：
1，要发送的数据**先存入**TWDR，在给**发命令**；
2，读操作**后再**从TWDR**获取**读取的数据

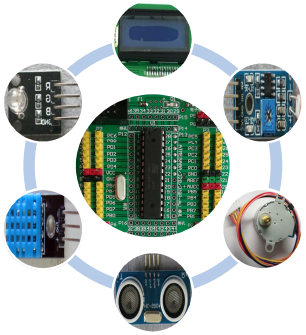


ATmega8A TWI地址寄存器-TWAR

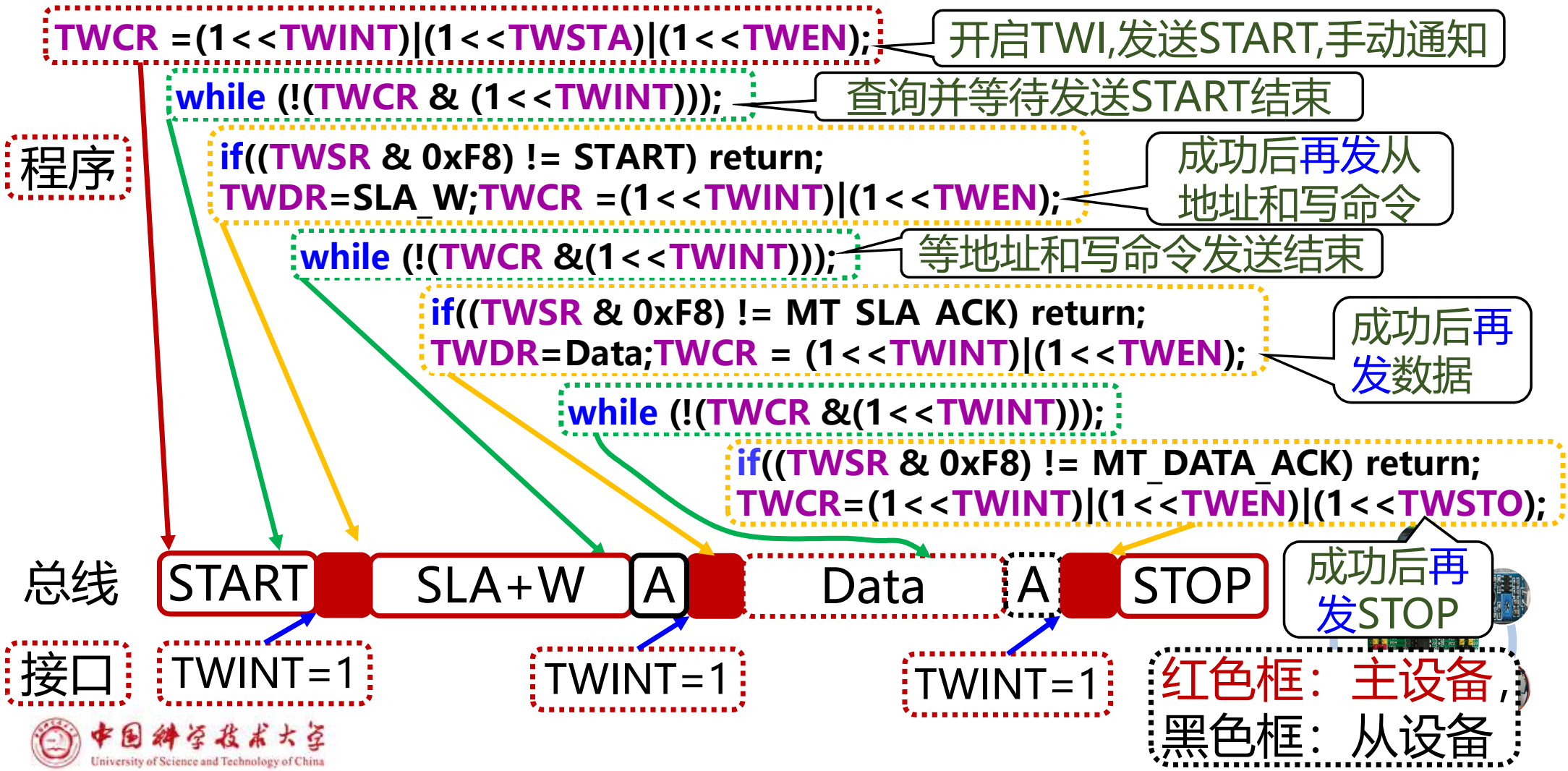
Bit	7	6	5	4	3	2	1	0
TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	0

TWI广播地址使能：为1时，接受广播地址

7位的从机地址，为从机时必须设置



TWI接口编程:主设备发送数据 (MT)



TWI接口编程:从设备接收数据(SR)

```
TWAR = sla addr;
TWCR =(1<<TWINT)|(1<<TWEA)|(1<<TWEN);
```

设置从设备地址，开启TWI、自动应答,手动通知

程序

```
while (!(TWCR &(1<<TWINT)));
```

等待主设备**寻址**自己

```
if((TWSR & 0xF8) != SR_SLA_ACK) return;
TWCR =(1<<TWINT)|(1<<TWEN)|(1<<TWEA);
```

主设备寻址自己后**等待接收**数据

```
while (!(TWCR & (1<<TWINT)));
```

从设备初始化后等待主设备寻址

```
if((TWSR & 0xF8) != SR_DATA_ACK) return;
Data = TWDR;
TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWEA);
```

总线 | START | SLA+W | A | Data | A | STOP

接口

TWINT=1

TWINT=1

红色框：主设备，
黑色框：从设备

TWI接口编程:从设备发送数据 (ST)

```
TWAR = sla addr;
TWCR =(1<<TWINT)|(1<<TWEA)|(1<<TWEN);
```

设置从设备地址，开启TWI、自动应答,手动通知

程序

```
while (!(TWCR & (1<<TWINT)));
```

等待主设备**寻址**自己

```
if((TWSR & 0xF8) != ST_SLA_ACK) return;
TWDR=Data;TWCR = (1<<TWINT)|(1<<TWEN);
```

主设备寻址自己后**发**送数据

从设备初始化后等待主设备寻址

```
while (!(TWCR &(1<<TWINT)));
```

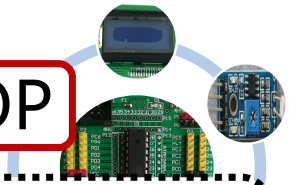
```
if((TWSR & 0xF8) != ST_DATA_ACK) return;
TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWEA);
```

总线

接口

TWINT=1

TWINT=1



红色框：主设备，黑色框：从设备

TWI接口编程:主设备接收数据 (MR)

```
TWCR =(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
while (!(TWCR &(1<<TWINT)));
if((TWSR & 0xF8)!=START) return;
TWDR=SLA_R;TWCR =(1<<TWINT)|(1<<TWEN);
while (!(TWCR &(1<<TWINT)));
if((TWSR&0xF8)!=MR_SLA_ACK) return;
TWCR =(1<<TWINT)|(1<<TWEN)|(1<<TWEA);
while (!(TWCR &(1<<TWINT)));
if((TWSR&0xF8)!=MR_DATA_ACK) return;
Data=TWDR;
TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
```

开启TWI,开启应答,发送START，手动通知

成功后再发SLA_R，手动通知

程序

总线

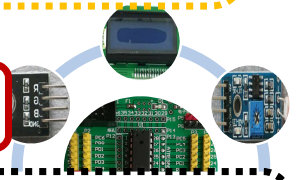
接口

START | SLA+R | A | Data | A | STOP

TWINT=1

TWINT=1

TWINT=1



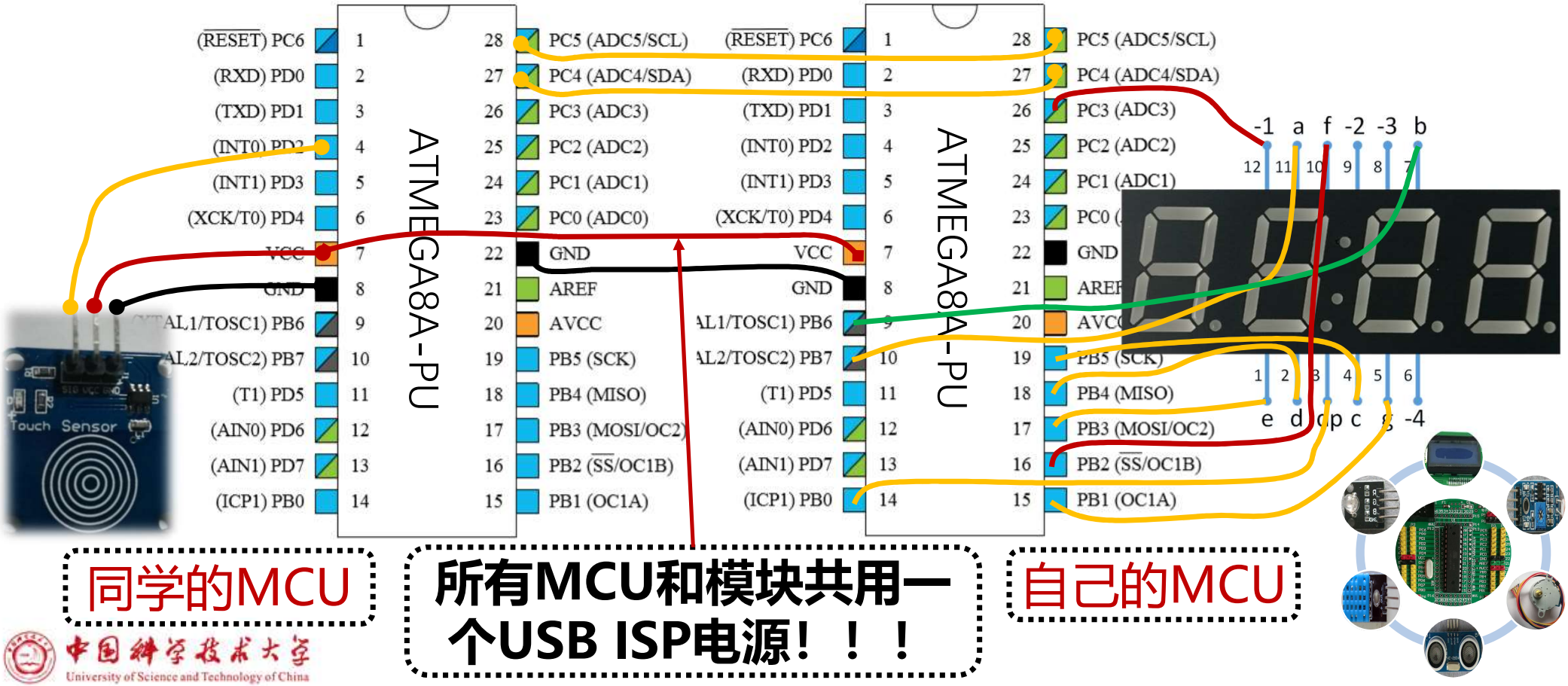
红色框：主设备，
黑色框：从设备

ATMEGA8A TWI编程示例

将一个MCU外接触摸开关的开关次数通过TWI显示在另一MCU外接的数码管上

TWI主设备发送

TWI从设备接收



同学的MCU

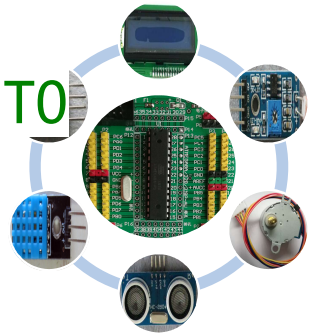
所有MCU和模块共用一个USB ISP电源！！！

自己的MCU

TWI主设备发送触摸开关数据的编程（1）

```
#include <avr/io.h>
#include <util/twi.h>  //TWI接口状态码定义等
#include <avr/interrupt.h>
unsigned char counter=0;
int main(void)
{ unsigned char sla_w = 0x33<<1;
  //从MCU地址为0x33(位0为开启广播)，写从MCU
 DDRC &= ~((1<<DDRC5)|(1<<DDRC4));//PC5/4为输入
 PORTC |= (1<<PORTC5)|(1<<PORTC4);//开启内部上拉
 TWBR = 0x02;//fscl=50KHz
 TWSR = 0x00;//无预分频
 DDRD &= ~(1<<DDRD2);//PD2(int0) 接触摸开关的sig管脚
 MCUCR |=((1<<ISC01)|(1<<ISC00));//int0管脚是上升沿触发中断INT0
 GICR |= (1<<INT0);//允许INT0外部中断
 sei(); //开启全局中断SREG(I)=1
```

```
ISR(INT0_vect)
{
    if(counter <15)
        counter++;
    else
        counter = 0;
}
```

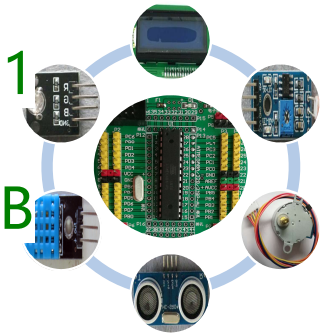


TWI主设备发送触摸开关数据的编程（2）

```
while (1)
{   TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);//清标志，开TWI，发START
    while(!(TWCR & (1<<TWINT)));//等待START发出
    if((TWSR & 0xf8) == TW  START)//START已发出
    {   TWDR=sla  w;//发送SLA+W
        TWCR=(1<<TWINT)|(1<<TWEN);//清除标志，并发送sla+w
        while(!(TWCR & (1<<TWINT)));//等待sla+w发出
        if((TWSR & 0xf8)==TW  MT  SLA  ACK)//sla+W已发出
        {   TWDR=counter;//发送开关次数
            TWCR=(1<<TWINT)|(1<<TWEN);//清除标志，并发送数据
            while(!(TWCR & (1<<TWINT)));//等待数据发出
            if((TWSR & 0xf8)==TW  MT  DATA  ACK)//数据已发出
            TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO);//清标志，发STOP
        }
    }
        TWCR=(1<<TWINT);//清除标志位，禁止TWI
 } //while结束
} //main函数结束
```

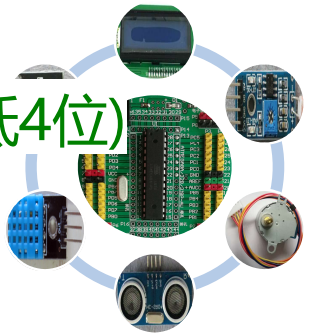
TWI从设备接收开关次数并显示的编程（1）

```
#include <avr/io.h>
#include <util/twi.h>  //TWI接口状态码定义等

int main(void)
{unsigned char counter=0;
 unsigned char sla_addr = 0x33<<1;//从机地址为0x33(最低位为开启广播)
 unsigned char seg7_hex[16]={0xfc,0x60,0xda,0xf2,0x66,0xb6,0xbe,0xe0,
    0xfe,0xf6,0xee,0x3e,0x9c,0x7a,0x9e,0x8e};//MSB=a,b,c,d,e,f,g,dp=LSB
 TWCR=0x0;//禁止TWI接口
 DDRC &=~((1<<DDRC5)|(1<<DDRC4));//PC5/4输入
 PORTC |=(1<<PORTC5)|(1<<PORTC4);//SCL/SDA内部上拉
 DDRC|=(1<<DDRC3);//PC3位输出
 PORTC &=~(1<<PORTC3);//PC3输出低电平给数码管的共阴极端 -1
 TWAR = sla_addr;//设置从机地址
 DDRB = (0xff); //端口B全为输出，控制数码管(MSB)a,b,c,...,dp=LSB
```



TWI从设备接收开关次数并显示的编程（2）

```
while (1)
{  TWCR = (1<<TWINT)|(1<<TWEA)|(1<<TWEN);//清标志，开TWI，自动应答
   while(!(TWCR & (1<<TWINT)));//等待接收sla+w
   if((TWSR & 0xf8)==TW_SR_SLA_ACK)//sla+W已收到，已发ACK
   {  TWCR = (1<<TWINT)|(1<<TWEA)|(1<<TWEN);//清除标志，开启ACK
      while(!(TWCR & (1<<TWINT)));//等待接收数据
      if((TWSR & 0xf8)==TW_SR_DATA_ACK)//数据已收到，已发ACK
      {   counter = TWDR; //收到的数据存储在counter里
          TWCR = (1<<TWINT)|(1<<TWEA)|(1<<TWEN);//
      }
   }
   TWCR = (1<<TWINT);//清除TWINT标志，关闭TWI接口
   PORTB= seg7_hex[counter & 0x0f]; //在七段数码管显示数据(仅低4位)
} //while结束
} //main函数结束
```

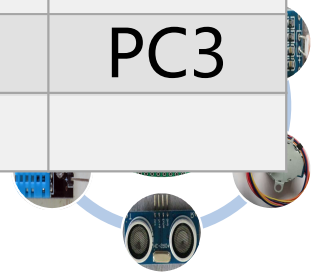


ATMEGA8A TWI编程示例的连线

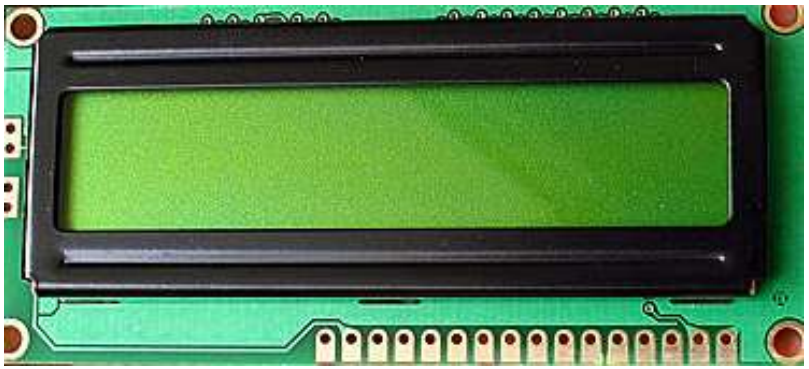
- 与同学合作，两块实验板，同学的板发触摸开关数据，自己的板接收数据并显示在数码管上
- 每次连接一个USB ISP 到电脑并下载对应的Hex文件
  - 千万**不要电源互通的**同时连接两个USB ISP下载线到电脑或电源
- 按下表连接MCU与各个模块

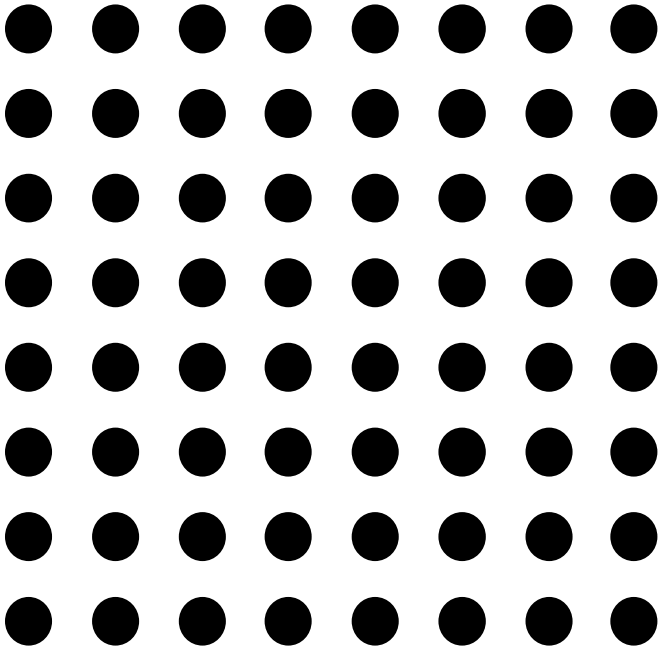
MCU1与TPAD和MCU2的连接			
触摸开关	**MCU1**	**MCU2**	**MCU1**
SIG	PD2	SCL	SCL
		SDA	SDA
		GND	GND
		VCC	VCC

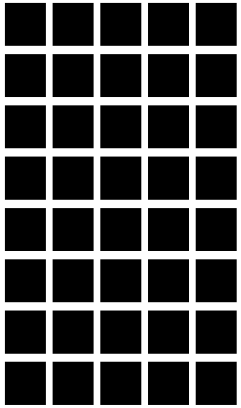
MCU2与数码管和MCU1的连接			
数码管	**MCU2**	**数码管**	**MCU2**
a	PB7	f	PB2
b	PB6	g	PB1
c	PB5	dp	PB0
d	PB4	-1	PC3
e	PB3		

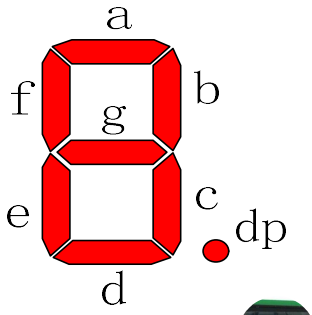


LCD1602液晶显示屏

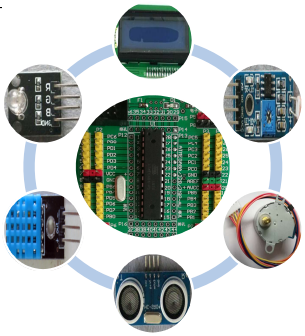




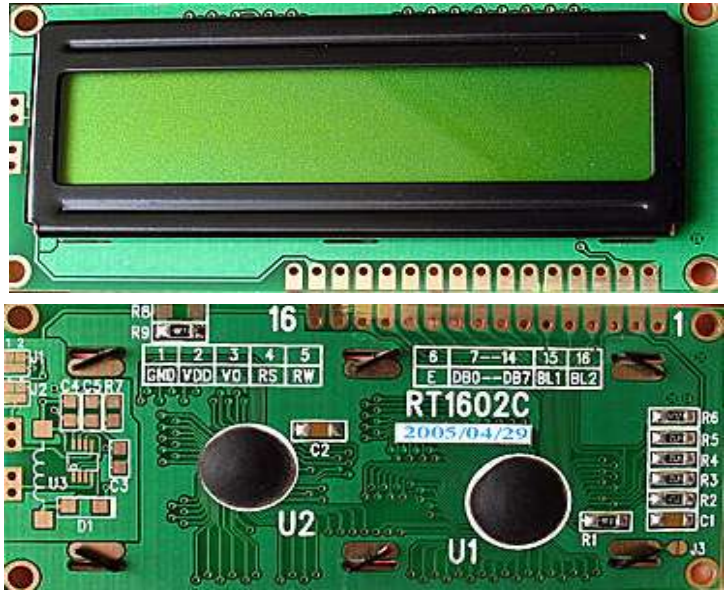




- ■点阵液晶模块
- ■可显示2*16字符
- ■每个字符的显示可由5×8或5×10点完成



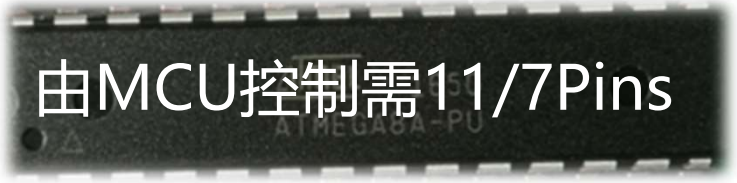
LCD1602实物与管脚

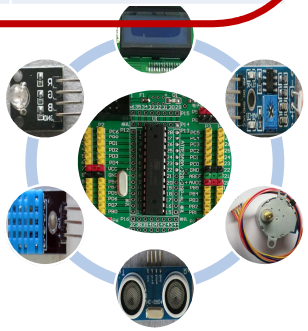


管脚	名称	说明	管脚	名称	说明
1	VSS	电源地	7	D0	数据IO 0
2	VDD	电源正极	8	D1	数据IO 1
3	V0	液晶显示偏压	9	D2	数据IO 2
4	RS	数据/命令选择	10	D3	数据IO 3
5	R/W	读/写信号	11	D4	数据IO 4
6	E	读写开始信号	12	D5	数据IO5
15	A	背光正极	13	D6	数据IO 6
16	K	背光负极	14	D7	数据IO 7

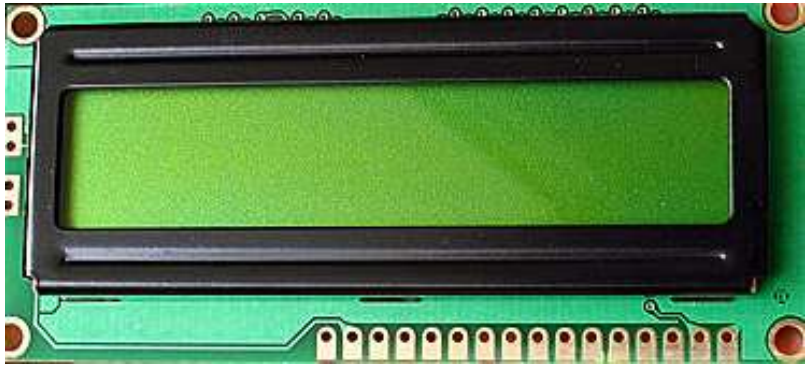
- 16Pins:电源,背光
- RS、RW、E和D[7:0]

支持8位或4位数据传输模式

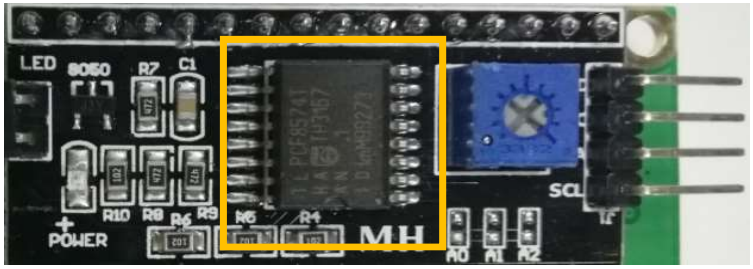




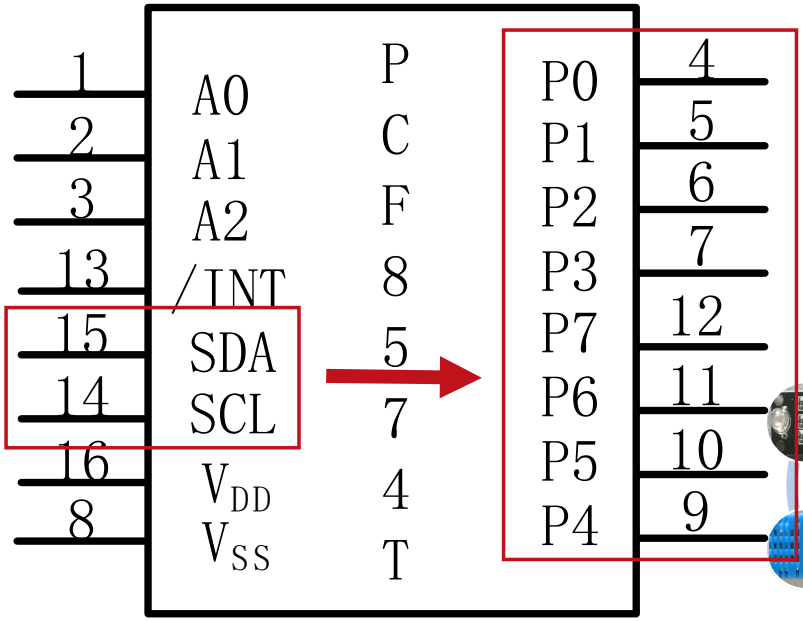
LCD1602接口的改进：减少接口连线



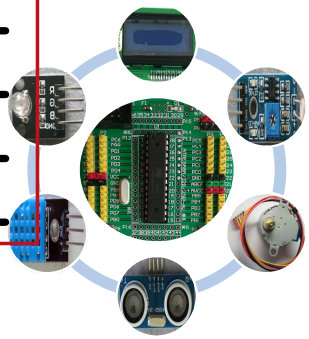
I²C串-8bits并



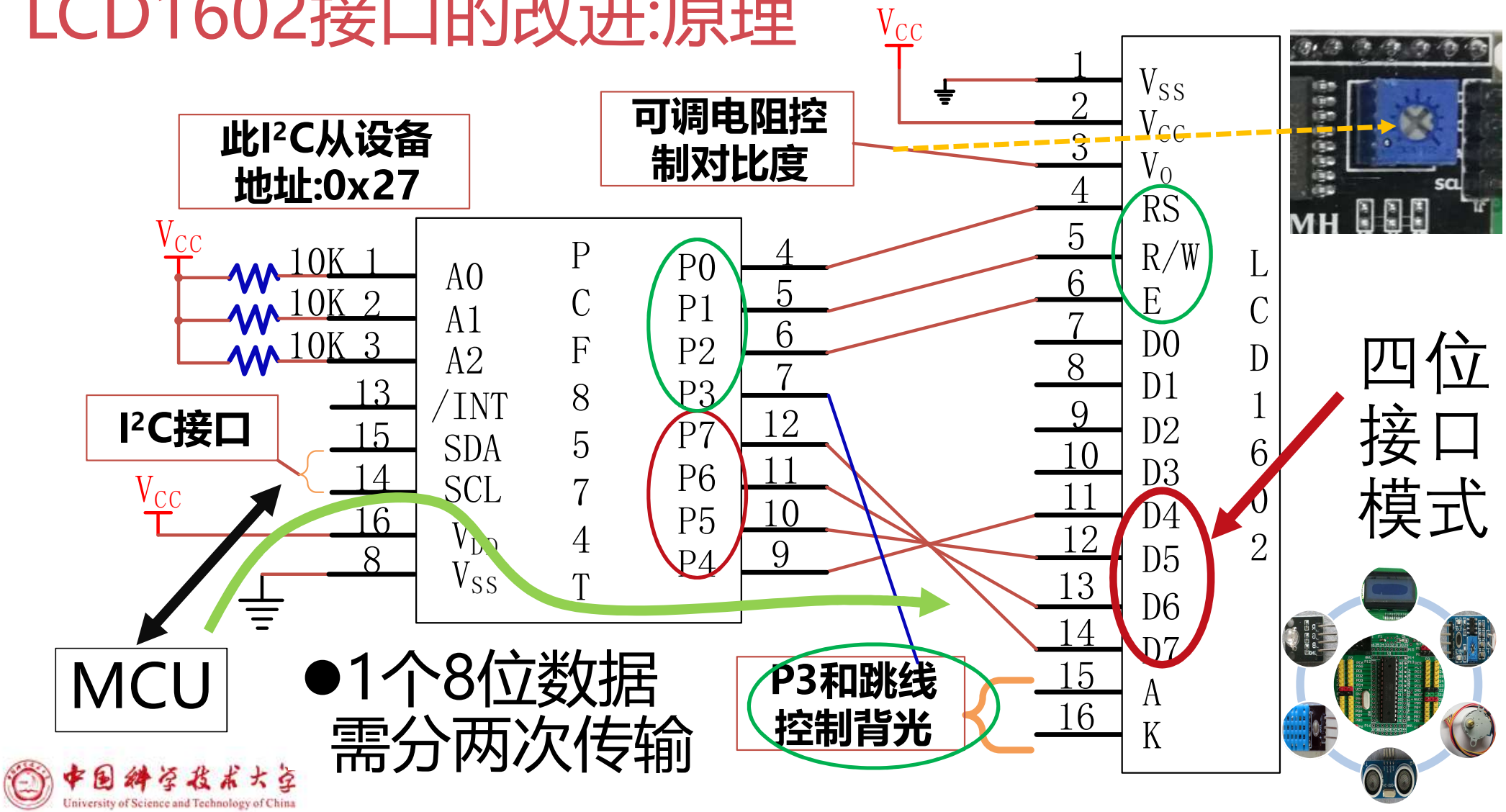
支持8位或**4位**数据传输模式



■MCU与LCD间仅需2Pins



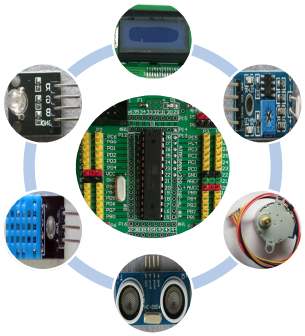
LCD1602接口的改进:原理



LCD1602液晶屏的显示控制

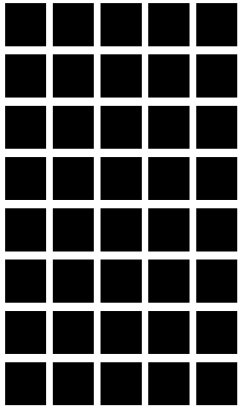
		1	2	3	4	5	6	…	40
DDRAM地址	行1	00h	01h	02h	03h	04h	05h	…	27h
	行2	40h	41h	42h	43h	44h	45h	…	67h

- 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)



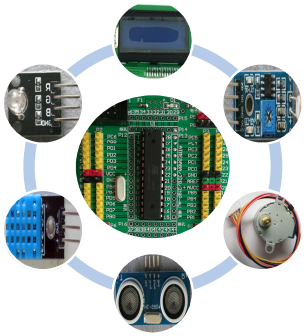
CGROM

前8个地址是可自定义的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	ﾞ	x̄
1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	ﾟ	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	ﾟ	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,2/1,5×8/10	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

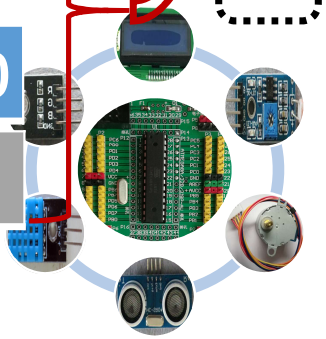
分2次发

■设置DDRAM地址指令

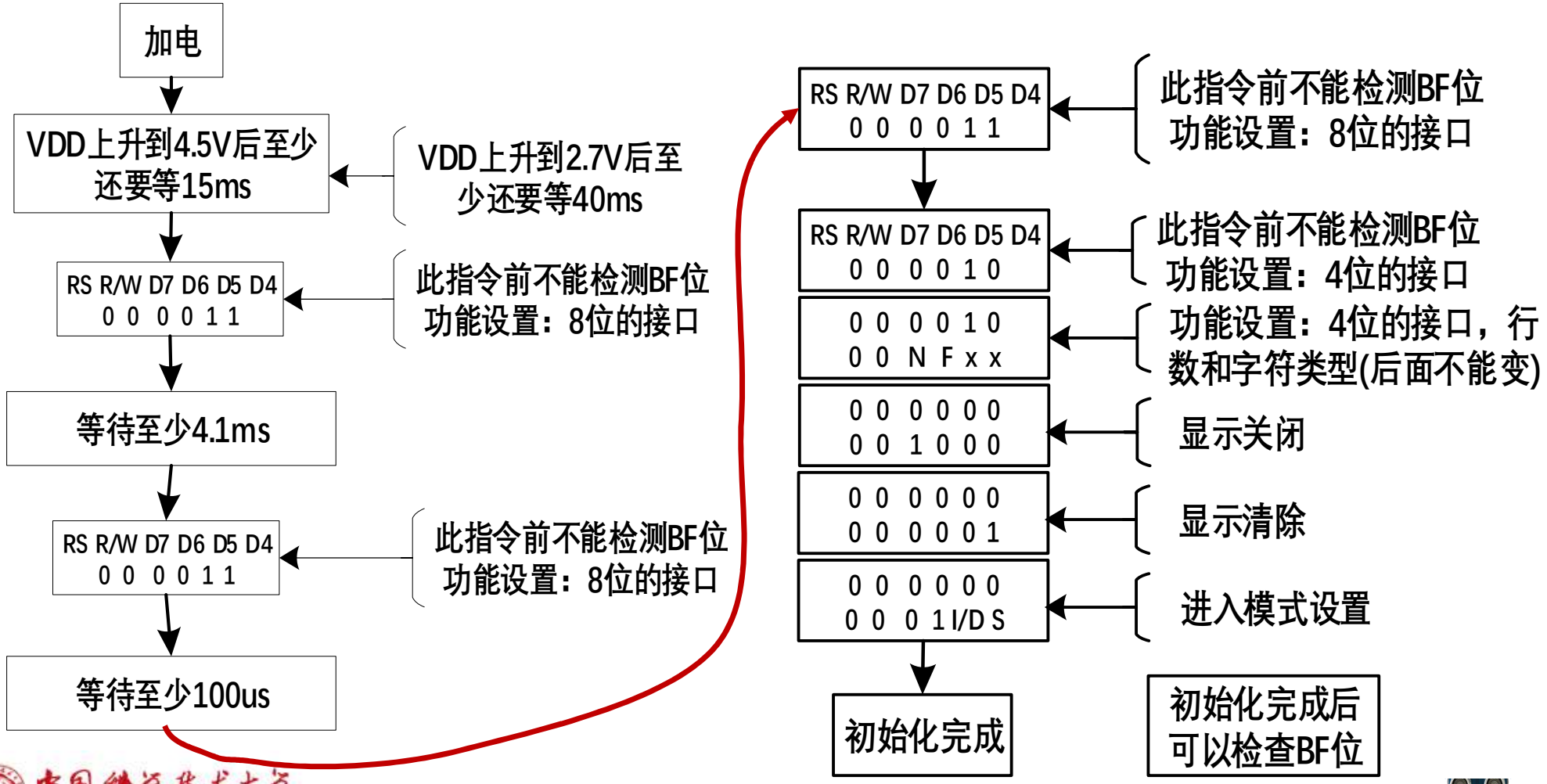
RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	A6	A5	A4	A3	A2	A1	A0

■向DDRAM写数据指令

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

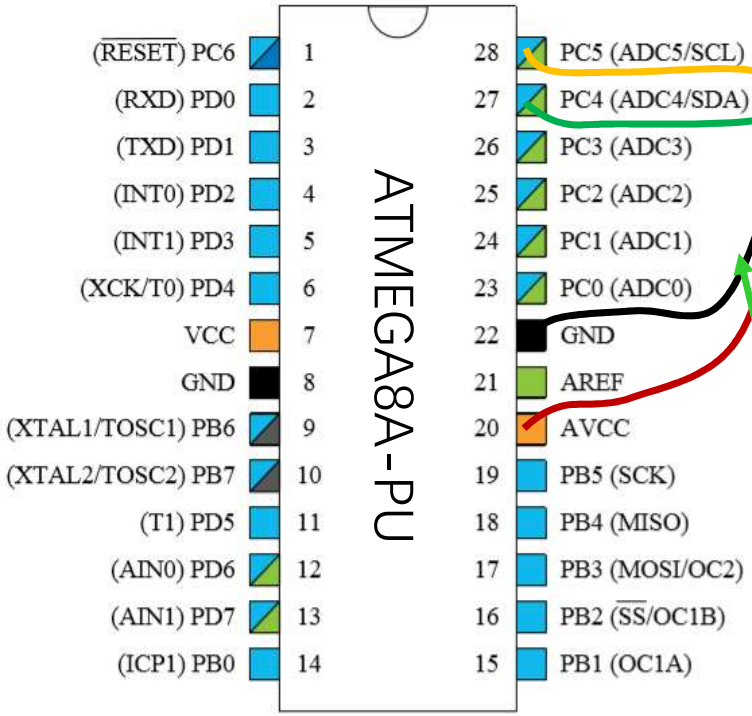


LCD1602的初始化：4位接口

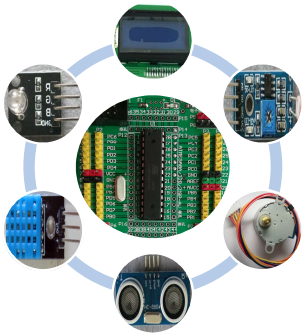


ATMEGA8A控制LCD1602显示实例

在LCD1602上显示字符



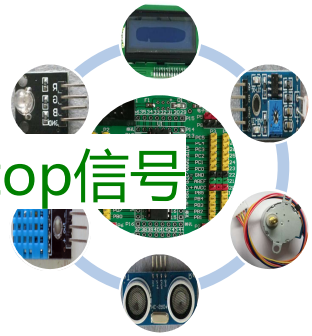
LCD1602与MCU
共电源正、负极



MCU控制LCD1602显示编程-"twi_fun.h”（1）

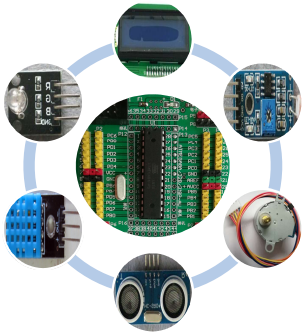
注：将TWI功能函数等写在一个头文件(twi_fun.h)里，方便后期使用

```
#include <util/twi.h>//软件自带TWI接口的寄存器等头文件
void TWI_Init(void)//twi 接口的初始化
{//设置SCL的频率：1MHz cpu-50KHz scl,2M-100K,8M-400K
    TWSR = 0x00; //最低2位为预分频设置(00-1,01-4,10-16,11-64)
    TWBR = 0x02; //位率设置，fscl=cpu频率/(16+2*TWBR*预分频值)
    TWCR = (1<<TWEN); //开启TWI
}
void TWI_Start(void)//发送Start信号，开始本次TWI通信
{ TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);//发送Start信号
  while(!(TWCR &(1<<TWINT)));//等待Start信号发出
}
void TWI_Stop(void)//发送Stop信号，结束本次TWI通信
{ TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);//发送Stop信号
}
```



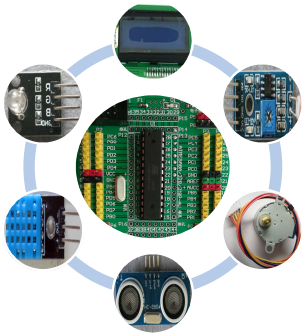
MCU控制LCD1602显示编程-"twi_fun.h”（2)

```
void TWI_Write(unsigned char uc_data) //向TWI接口发送8位数据
{
    TWDR = uc  data;//8位数据存放在TWDR
    TWCR = (1<<TWINT)|(1<<TWEN);//发送TWDR中的数据
    while(!(TWCR &(1<<TWINT)));//等待数据发出
}

unsigned char TWI_Read_With_ACK(void)
{
    TWCR = (1<<TWINT)|(1<<TWEA)|(1<<TWEN);//准备接收数据，并
    ACK
    while(!(TWCR &(1<<TWINT)));//等待接收数据
    return TWDR;//返回接收到的数据
}
```



MCU控制LCD1602显示编程-"twi_fun.h”（3）

```
unsigned char TWI_Read_With_NACK(void)
{
    TWCR = (1<<TWINT)|(1<<TWEN);//准备接收数据，并
    NACK
    while(!(TWCR &(1<<TWINT)));//等待接收数据
    return TWDR;//返回接收到的数据
}
unsigned char TWI_Get_State_Info(void)
{
    unsigned char uc  status;
    uc  status = TWSR & 0xf8;
    return uc_status;
}
```

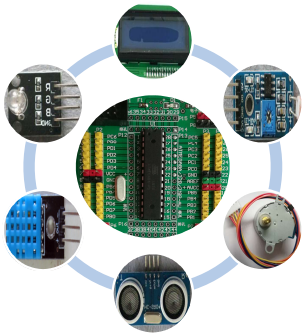


MCU控制LCD1602显示编程-"twi_lcd.h"(1)

```
#ifndef F_CPU
#define F_CPU 1000000UL  //延时用
#endif
#include "twi_fun.h"
#include <util/delay.h>
```

用TWI函数，发送LCD命令与数据等，写到另一个头文件(twi_lcd.h)里，以方便后期用

```
//LCD1602 控制和显示指令
#define LCD_CLEARDISPLAY 0x01//清屏，设置AC为DDRAM地址0
#define LCD_RETURNHOME 0x02//设置AC为DDRAM地址0，光标回原点
#define LCD_ENTRYMODESET 0x04//与I/D和S位定义光标移动方向和显示移位
#define LCD_DISPLAYCONTROL 0x08//与D/C/B设置显示开关，光标开关/闪烁
#define LCD_CURSORSHIFT 0x10//与S/C和R/L一起设置光标移动或显示移位
#define LCD_FUNCTIONSET 0x20//与DL、N和F一起设置LCD功能:8/4位数据;1/2行显示;5*8/10点阵字符
#define LCD_SETCGRAMADDR 0x40//设置CGRAM地址到地址计数器(AC)
#define LCD_SETDDRAMADDR 0x80//设置DDRAM地址到地址计数器(AC)
```

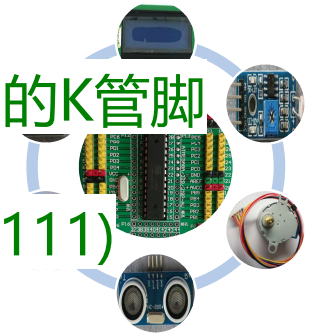


MCU控制LCD1602显示编程-"twi_lcd.h"(2)

```
// LCD进入模式设置位(LCD_ENTRYMODESET=0x04)
#define LCD_ENTRYSHIFT 0x01//S位=1，显示移位，=0不移位
#define LCD_ENTRYINC 0x02//I/D位=1，显示左移(递增)

//LCD显示开关控制位 (LCD_DISPLAYCONTROL=0x08)
#defineLCD_BLINKON 0x01//B=1，闪烁
#define LCD_CURSORON 0x02//C=1，光标
#define LCD_DISPLAYON 0x04//D=1，显示开

//LCD光标和显示移位控制位(LCD_CURSORSHIFT=0x10)
#define LCD_CURSOR2LEFT 0x00//S/C=0,R/L=0:光标往左移
#define LCD_CURSOR2RIGHT 0x04//S/C=0,R/L=1:光标往右移
#defineLCD_DC2LEFT 0x08//S/C=1,R/L=0:显示向左移，光标跟着移
#define LCD_DC2RIGHT 0x0C//S/C=1,R/L=1:显示向右移,光标跟着移
```

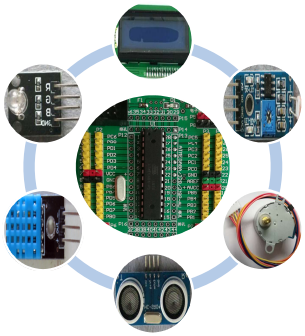
MCU控制LCD1602显示编程-"twi_lcd.h"(3)

```
//LCD功能设置位(LCD_FUNCTIONSET=0x20)
#define LCD_4BITMODE 0x00  //DL=0:4位(DB7-4)数据，需2次传输
#define LCD_8BITMODE 0x10  //DL=1：8位(DB7-0)数据传输
#define LCD_1LINE 0x00  //N=0，1行显示
#define LCD_2LINE 0x08  //N=1，2行显示
#define LCD_5X8DOTS 0x00  //F=0：5X8 dots字符
#define LCD_5XADOTS 0x04  //F=1：5X10 dots字符，只能1行显示
//LCD 1602 控制管脚：I2C数据的低4位（PCF8574-P0~3)
#define LCD_RS 0x01  //PCF8574-P0控制LCD1602的RS管脚
#define LCD_RW 0x02  //PCF8574-P1）控制LCD1602的RW管脚
#define LCD_E 0x04  //PCF8574-P2）控制LCD1602的E管脚
#define LCD_BACKLIGHTON 0x08  //PCF8574-P3控制LCD1602的K管脚

#define LCD_SLAVE_ADDRESS 0x27  //从机地址PCF8574(A2-0:111)
```



MCU控制LCD1602显示编程-"twi_lcd.h"(4)

```
unsigned char TWI_Write_LCD(unsigned char uc_data)
{ TWI_Start();//发送START信号
 if(TWI_Get_State_Info()!=TW_START) return 0;//不成功

 TWI_Write(LCD_SLAVE_ADDRESS<<1|TW_WRITE); //发送SLA+W
 if(TWI_Get_State_Info()!=TW_MT_SLA_ACK)return 0;//不成功

 TWI_Write(uc_data|LCD_BACKLIGHTON);//发送数据+背光常开
 if(TWI_Get_State_Info()!=TW_MT_DATA_ACK)return 0;//不成功

 TWI_Stop();
 return 1;//成功
}
```



MCU控制LCD1602显示编程-"twi_lcd.h"(5)

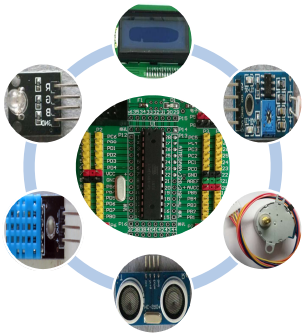
```
void LCD_4Bit_Write(unsigned char uc_data)//4位方式写PCF8574
{ TWI_Write_LCD(uc_data);//数据送出，E=0
    delay_us(1);//保持
   TWI_Write_LCD(uc_data|LCD_E);//数据送出，E=1
    delay_us(1);//保持
   TWI_Write_LCD(uc_data & (~LCD_E));//数据送出，E=0
   _delay_us(50);//等待数据传输结束
}
void LCD_8Bit_Write(unsigned char uc_data,unsigned char uc_mode)
  //2次4位数据传输方式写PCF9574,uc_mode:0-命令,1-数据
{  unsigned char high4bit = uc_data & 0xf0;
    unsigned char low4bit = (uc_data<<4)&0xf0;
    LCD_4Bit_Write(high4bit|uc_mode);//先发送高4位
    LCD_4Bit_Write(low4bit|uc_mode);//再发送低4位
}
```

MCU控制LCD1602显示编程-"twi_lcd.h"(6)

```
void LCD_Init()//初始化LCD1602
{  _delay_ms(50);//上电后至少再等40ms
    LCD_4Bit_Write(0x30); //在默认8位接口，试着进入4位接口模式
     delay_us(4500);//等待至少4.5ms
    LCD_4Bit_Write(0x30);
     delay_us(4500);//等待至少4.5ms
    LCD_4Bit_Write(0x30);
     delay_us(150);//等待至少150us
    LCD_4Bit_Write(0x20);//进入4位接口模式
    //设置模式，显示，点数等
    LCD_8Bit_Write(LCD_FUNCTIONSET|LCD_4BITMODE|LCD_2LINE|LCD_5X8DOTS,
    0);
    LCD_8Bit_Write(LCD_DISPLAYCONTROL|LCD_DISPLAYON,0);//显示
    LCD_8Bit_Write(LCD_CLEARDISPLAY,0);
     delay_us(2000);//等待
    LCD_8Bit_Write(LCD_ENTRYMODESET|LCD_ENTRYINC,0);//显示左移(递增)
    LCD_8Bit_Write(LCD_RETURNHOME,0);//返回原点
    _delay_us(2000);//等待
}
```

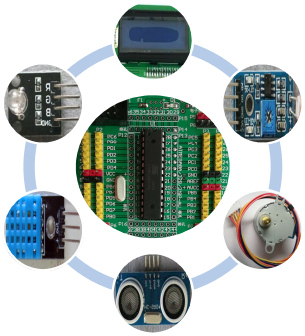
MCU控制LCD1602显示编程-"twi_lcd.h"(7)

```
void LCD_Set_Cursor_Location(unsigned char row,unsigned char col)
//设置光标位置,row:0~1,col:0~39
{ unsigned char offset[]={0x0,0x40};
  LCD_8Bit_Write(LCD_SETDDRAMADDR|(col+offset[row]),0);
}
void LCD_Write_NewChar(char c_data)//在当前位置显示
{  LCD_8Bit_Write(c_data,1);
}
void LCD_Write_Char(unsigned char row,unsigned char col,char c_data)
//在指定位置显示
{ LCD_Set_Cursor_Location(row,col);
  LCD_8Bit_Write(c_data,1);
}
```



MCU控制LCD1602显示编程-"twi_lcd.h"(8)

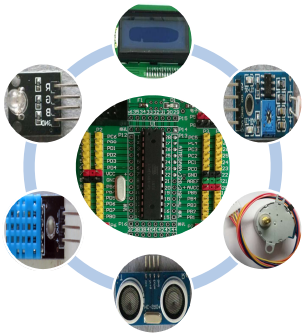
```
void LCD_Write_String(unsigned char row,unsigned char col,const char *pStr)
//在指定位置显示串
{
    LCD_Set_Cursor_Location(row,col);
    while((*pStr) != '\0')
    {
        LCD_8Bit_Write(*pStr,1);
        pStr ++;
    }
}
```



MCU控制LCD1602显示编程-主程序

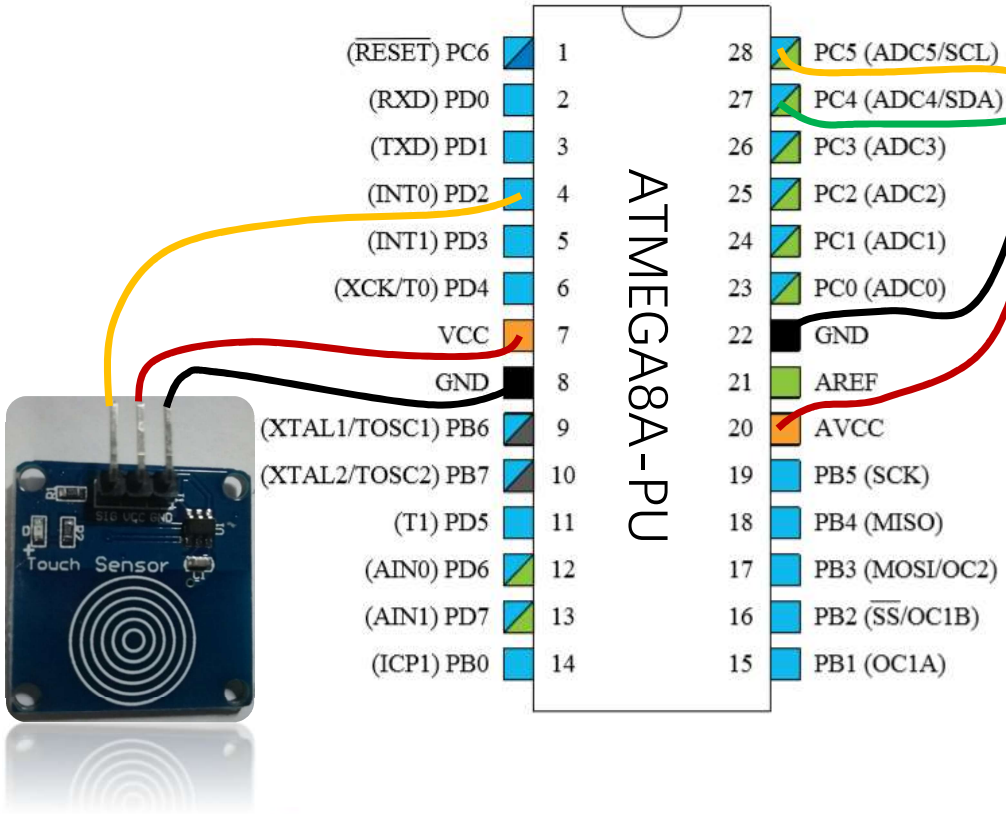
```
#include <avr/io.h>
#include "twi_lcd.h"
int main(void)
{
    TWI_Init();
    LCD_Init();

   while (1)
   {
        LCD_Write_String(0,3,"Welcome to");
        LCD_Write_String(1,3,"USTC.");
   }
}
```

MCU与LCD1602的连接	
MCU	**LCD1602**
SCL	SCL
SDA	SDA
GND	GND
VCC	VCC



MCU控制LCD1602显示实例2

用中断方式统计触摸开关的开关次数并显示在LCD1602上

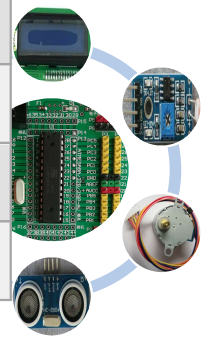


MCU与LCD1602的连接	
MCU	**LCD1602**
SCL	SCL
SDA	SDA
GND	GND
VCC	VCC

MCU控制LCD1602显示实例2 的编程核心

```
while (1)
{   d_t = counter;
    for(i=0;i<4;i++)//1个整数转换为4位字符
    {
     chs[3-i]=d_t%10+0x30;
     d_t=d_t/10;
    }
    LCD_Write_String(0,3,"Touchpad Times:");
    LCD_Write_String(1,8,chs);
}
```

MCU与触摸开关的连接	
MCU	**触摸开关**
PD2	SIG
GND	GND
VCC	VCC



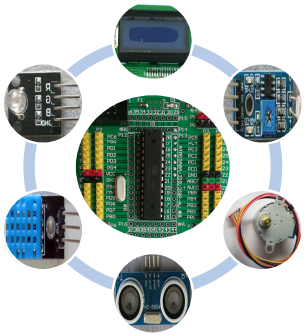
本周实验内容

实验内容1：在LCD1602上显示本人姓名（拼音）和学号

实验内容2：设计**一**个程序**分别**下载到自己和同学的MCU中，实现将一块板上触摸开关的开关**次数**，通过**TWI**接口传送**到**在另一块板并显示在其LCD上。满足以下之一即可：

- 单独设计收和发程序，借用同学的电路板完成
- 单独设计收和发程序，两个同学合作完成
- 设计**收发**一体的程序，借用电路板或合作完成

当次全部完成后当场验收，总结下次交



实验注意事项与应急处理

- 连接下载线到电脑USB接口时应注意：
  - 实验板电源指示灯是否亮，如不亮可能存在连接短路，须立即断开下载线与电脑USB接口的连接
- 打开下载程序PROGISP（Ver 2.0）
  - 确认下载线连接正确（连接状态图标为彩色）
  - 确认芯片型号正确，调入Flash程序正确等