51 单片机

目 录

串口通信

是一种通讯接口，可实现两设备的互相通信。

单片机的串口可使单片机与单片机、单片机与电脑（USB 接口，串口助手）、单片机与各种模块互相通信。狼牙串口，可以实现手机与单片机通信。

51 单片机内部自带 UART（通用异步收发器）可实现单片机的串口通信。

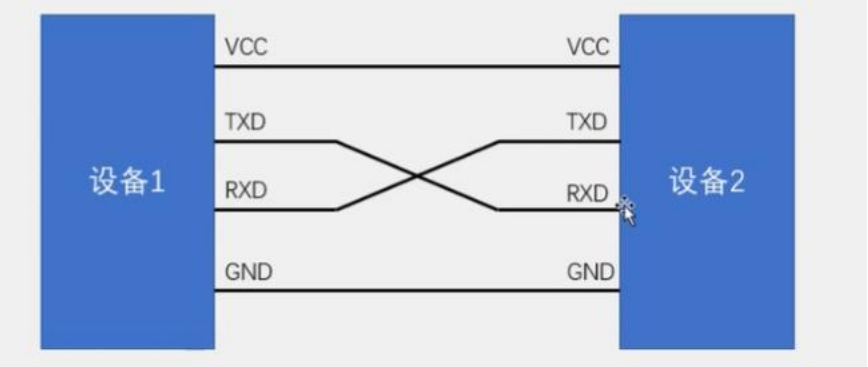
硬件电路

简单双向串口通信有两根通信线，发送端 TXD(Transmit)和接收端 RXD(Receive)。

TXD 与 RXD 交叉连接（接化发）。

单向数据传输时，可以直接一根通信线。

当电平标准不一致时，需要加电平转换芯片。



设备可独立供电时，不需要接 VCC。

电平标准

电平标准是数据 1 和数据 0 的表达方式，是传输电缆中人为规定的电压与数据的对应关系。

TTL 电平：+5V 表示 1，0V 表示 0；

RS232 电平：-3~-15V 表示 1，+3~+15V 表示 0；（前两者都是相对于 GND 的压差）

RS485 电平：两线压差+2~+6V 表示 1，-2~-6V 表示 0（差分信号）。

常见通信接口

名称	引脚定义	通信方式	特点
UART	TXD、RXD	全双工、异步	点对点通信
I^2C	SCL、SDA	半双工、同步	可挂载多个设备
SPI	SCLK、MOSI、MISO、CS	全双工、同步	可挂载多个设备
1-Wire	DQ	半双工、异步	可挂载多个设备

此外还有 CAN(差分信号)、USB 等。

全双工：通信双方可以在同一时刻互相传输数据。

半双工：通信双方可以互相传输数据，但必须分时复用一根数据线收发。

单工：通信只有一方发送到另一方，不能反向传输。

异步：通信双方各自约定通信速率（各自设定采样间隔）。

同步：通信双方靠一根时钟线如 SCL、SCLK 来约定通信速率（上升沿采样）。

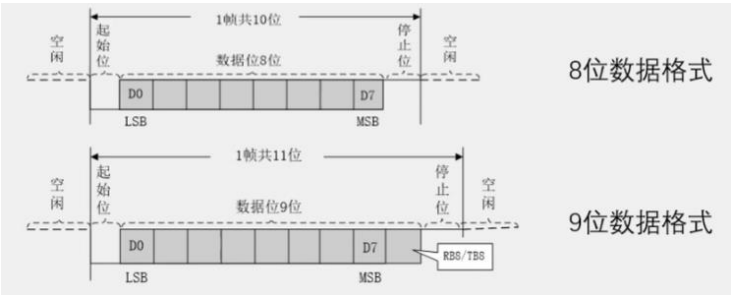
总线：连接各个设备的数据传输线路（类似于一条马路，把路边各住户连接起来，使住户可以相互交流），I^2C 总线、SPI 总线、1-Wire 总线等。

UART 串口参数及时序图

波特率：串口通信的速率（发送和接收各数据位的间隔时间）

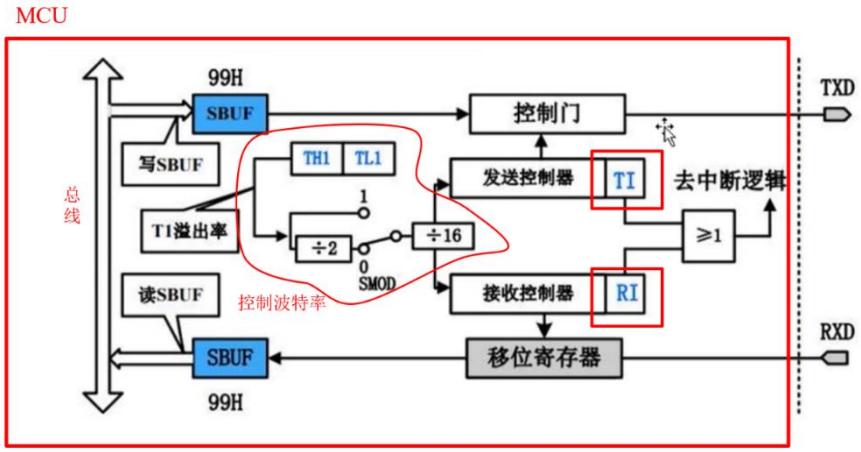
[比特率对应字节]

检验位：用于数据检验（奇偶检验，始终保证序列的 1 数量为奇数或偶数）



串口模式图

SBUF：串口数据缓存寄存器，两个独立寄存器，但占用相同的地址。



定时器

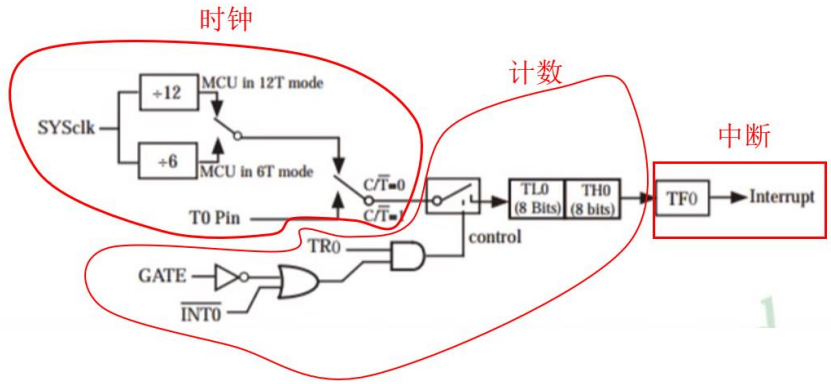
根据时钟的输出信号，每隔一定间隔，计数单元的数值就增加一，当计数单元数增加到“设定的提醒时间”，计数单元就会向中断系统发出中断申请，产生提醒，使程序跳转到中断服务函数中执行。

工作模式

STC89C52 的 T0 和 T1 均有四种工作模式：

模式 0：13 位定时器/计数器

模式 1：16 位定时器/计数器（常用）



计数系统中含两个字节 TH 和 TL，可以存 65535 个数。

模式 2：8 位自动重装模式

模式 3：两个 8 位计数器

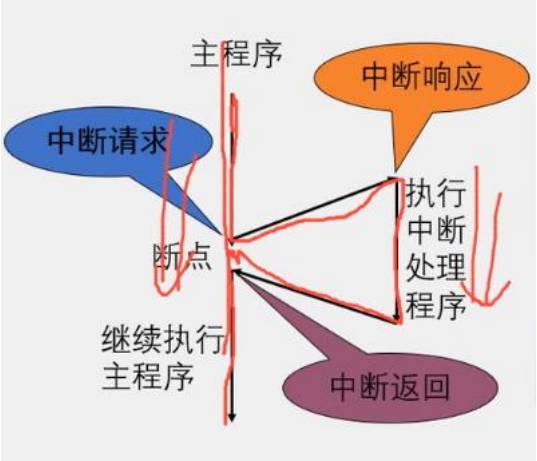
中断系统

CPU 总是先响应优先级别最高的中断请求。

中断嵌套：当 CPU 正处理一个中断源请求时，发生一个优先级更高的中断源请求，如果 CPU 能暂停对原来中断源的服务程序，转而去处理优先级更高的中断请求源，处理完后再回到原低级中断服务程序。

中断程序的编写与普通子函数差别不大，但需在函数名后跟上中断号，如 Interrupt0、Interrupt1 等。

在主程序中，先进入中断子程序执行后再返回主程序。



定时器相关寄存器配置

两个常见寄存器：

TCON，定时器 T1、T0 的控制寄存器，可位寻址；

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	name	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF，定时器溢出标志，当最高位产生溢出时由硬件置“1”，向 CPU 请求中断。

TR，定时器运行控制位，如 TR1=0 时禁止 T1 计数，TR1=1 时就允许 T1 开始计数。

TMOD，定时器模式寄存器，不可位寻址（只能整体赋值）。

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
定时器1				定时器0			

M1 和 M0，计数器模式选择。

GATE，GATE=0 时，TR 单独控制定时器运行；GATE=1 时，只有在外部引脚 $\overline{INT}$ 为高且 TR 为 1 时才打开定时器。

当同时配置定时器 1 和定时器 0 时，整体赋值可能会使配置好的定时器状态被刷新。

```
// TMOD=0x0001;    //0000 0001
TMOD=TMOD&0xF0; //把TMOD拉回低四位清0，高四位保持不变
TMOD=TMOD&0x01;//把TMOD拉回低四位置1，高四位保持不变
```

中断寄存器（IE、IP）：

IE(Interrupt Enable):

EA，CPU 总中断允许控制位；ET，溢出中断允许位；

IP(Interrupt priority low):

PT 中断优先级控制位。

LED 点阵屏

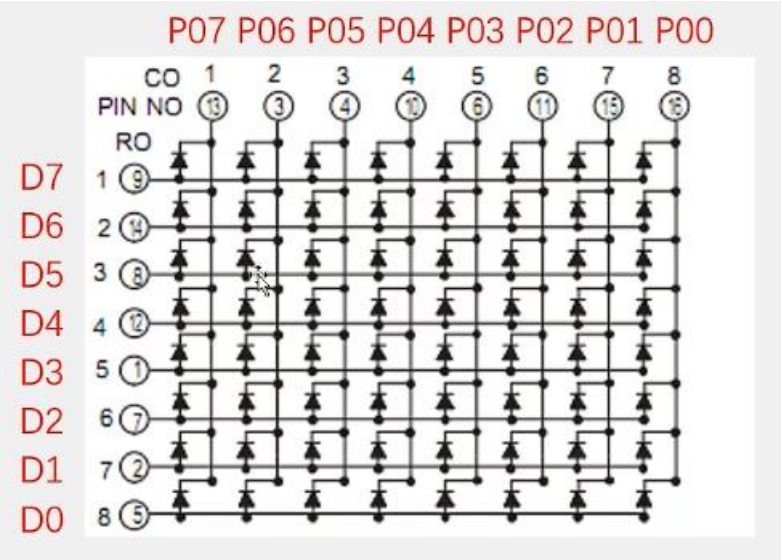
工作原理

LED 点阵屏的结构类似数码管，只不过以每一列像素以 8 字型排列。

LED 点阵屏与数码管一样，有共阴极和共阳极两种接法，不同的接法对应的电路结构不同（阳极选中置 1，阴极选中置 0）。

LED 点阵屏需要进行逐行或逐列扫描，才能使所有 LED 同时显示。

开发板引脚及对应关系，D0~D4 接到 74HC595 上。



74HC595

74HC595 是串行输入并行输出的移位寄存器，可用 3 根线输入串行数据，8 根线输出并行数据，多片级联后，可输出 16 位、24 位、32 位等，常用于 IO 口扩展。

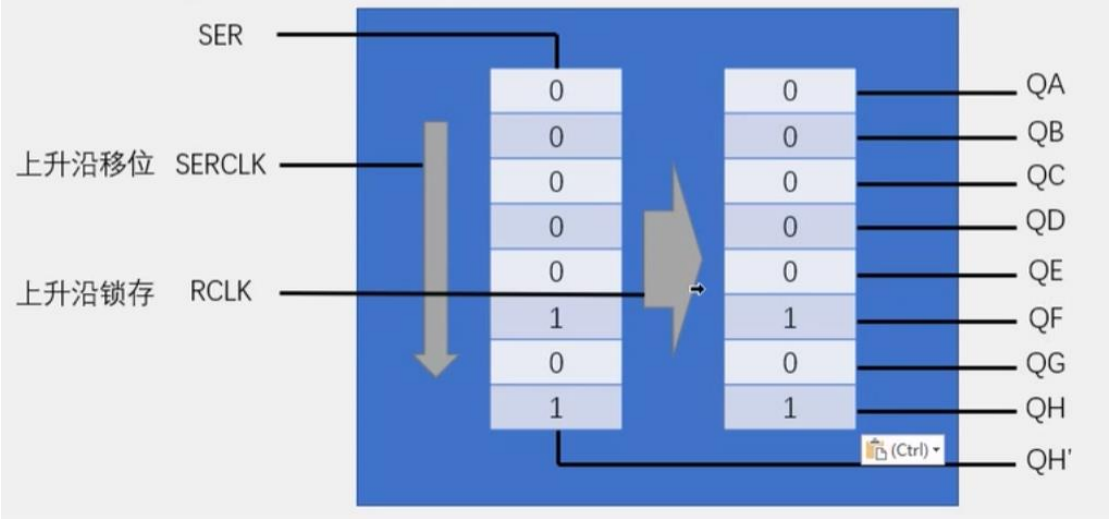
OE 为输出使能，低电平有效。对应的，JOE 上 OE 与 GND 短接。

OA~OH 为 8 个并行输出，OH’用于多片级联。

SER 接入串行数据。

SERCLK 串行时钟，每遇上升沿，数据向下移位。

RCLK 寄存器时钟，上升沿锁存。



当满 8 位后，SER 继续输入，若 QH’接下一个移位寄存器，被挤出的数将进入新的移位寄存器的输入缓存。

C51 的 sfr 和 sbit

sfr，特殊功能寄存器声明。例如 sfr P0 = 0x80;声明 P0 口寄存器的物理地址为 0x80；

sbit，特殊位声明。例如 sbit P0_1=0x81;声明 P0 寄存器的第一位。

对不可位寻址的寄存器，若只要操作其中一位而不影响其它位时，可用“&=”（与等于）、“|=”（或等于）、“^=”的方法进行位操作。

“&=”，一般是对某一位进行清零。例如对第 0 位进行清零，则将其与 0xFE（11111110）进行&=。

“|=”,一般是对某一位进行置 1。例如对第 1 位进行置 1，则将其与 0x01(00000001)进行相或。

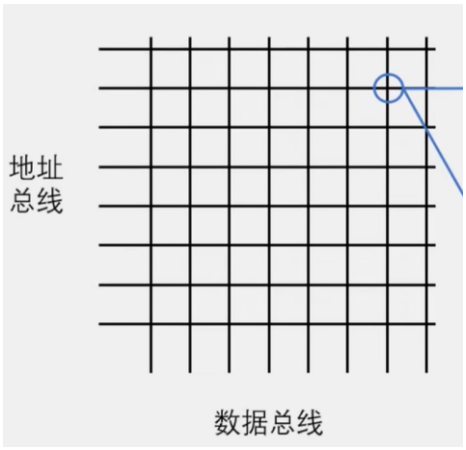
“^=”,一般是对某一位进行取反。

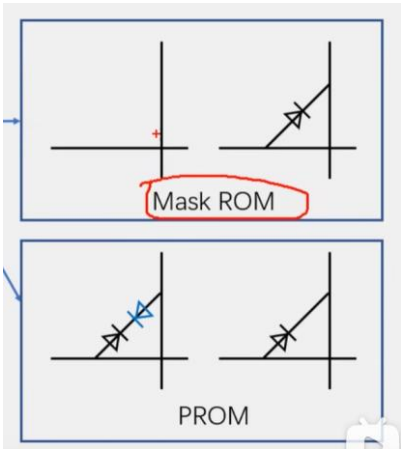
AT24C02

存储器介绍

易失性存储器 RAM，掉电丢失数据。

非易失性存储器 ROM，存储速度较慢。PROM 只能写入一次，EPROM 可擦除可编程，E2PROM，电可擦除可编程。Flash（闪存）。硬盘、软盘、光盘等。





横向为地址总线，纵向为数据总线，类似于行列扫描，控制节点短路或断路。

在 Mask ROM 中，存储断开就啥都不接，进行存储时为避免每一行互不干扰接二极管，电路不可改变。

在 PROM 中，可以通过击穿蓝色的二极管进行导通（烧录）。还有熔丝型来控制编程，未烧断时短路，加大电流烧断后断路。

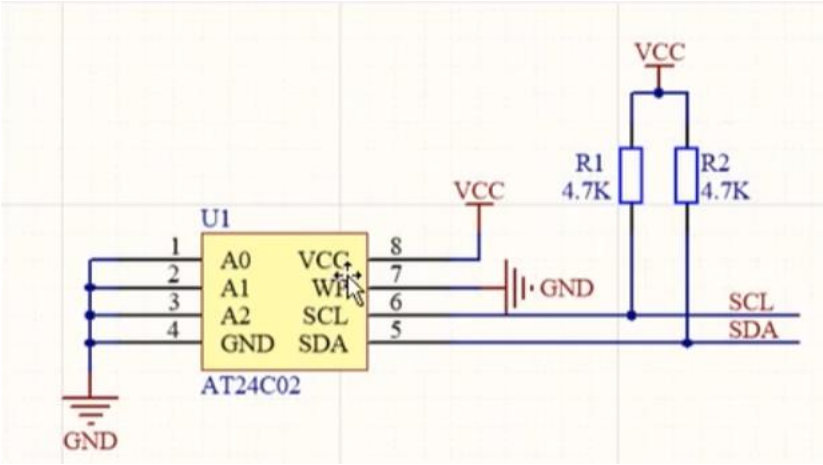
AT24C02 介绍

是一种掉电不丢失的存储器，可用于保存单片机运行时想要永久保存的数据信息。

存储介质：E2PROM

通讯接口：I2C 总线

容量：256 字节



引脚	功能
VCC、GND	电源（1.8V～5.5V）
WP	写保护（高电平有效）
SCL、SDA	I2C 接口
A0、A1、A2	I2C 地址

I2C 总线

一种通用数据总线。

包含两根通信线：SCL、SDA（串行数据线）。

同步、半双工、带数据应答。

所有 I2C 设备的 SCL 连在一起，SDA 连在一起。

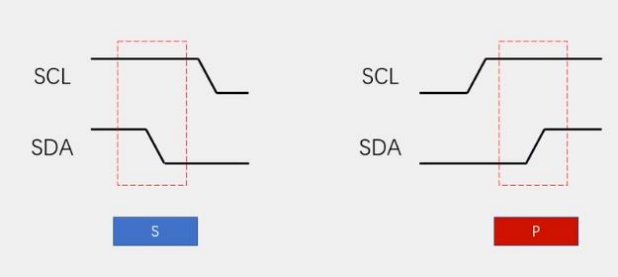
设备的 SCL 和 SDA 均要配置成开漏输出模式。（浮空状态，电路断开，电压不稳定）

SCL 和 SDA 各添加一个上拉电阻，阻值一般为 4.7KΩ。

开漏输出和上拉电阻的共同作用实现了“线与”功能，此设计解决了多机通信互相干扰的问题。

为保证通信时不受干扰，其他引脚全部输出 1，处于断开状态。

I2C 时序结构



起始条件：SCL 高电平期间，SDA 从高电平切换到低电平

```
void I2C_Start(void)
{
    I2C_SDA=1;
    I2C_SCL=1;
    I2C_SDA=0;
    I2C_SCL=0;
}
```

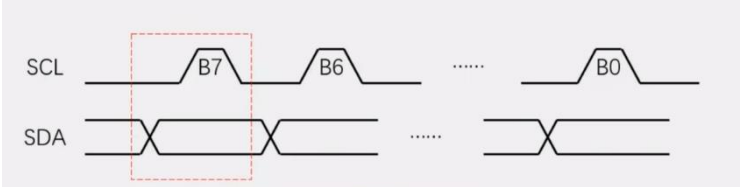
为确保任何情况下（可能在先发送再接收数据帧中应用）SCL 和 SDA 都是高电平，先令他们置 1。

终止条件：SCL 高电平期间，SDA 从低电平切换到高电平

```
void I2C_Stop(void)
{
    I2C_SDA=0;
    I2C_SCL=1;
    I2C_SDA=1;
}
```

用在发送一帧数据或接收一帧数据时，SDA 可能是 0 或 1，所以先令 SDA 置 0。

发送一个字节：



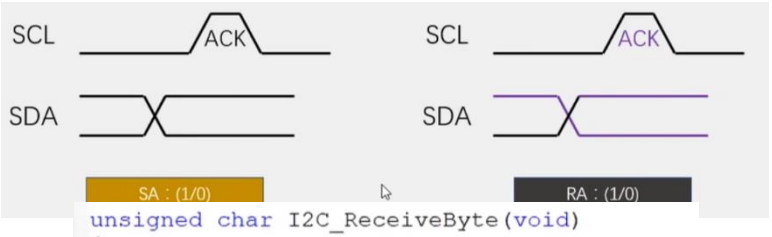

```
void I2C_SendByte(unsigned char Byte)
{
    unsigned char i;
    for(i=0;i<8;i++)
    {
        I2C_SDA=Byte&(0x80>>i);
        I2C_SCL=1;
        I2C_SCL=0;
    }
}
```

SCL 低电平期间，主机将数据位依次放到 SDA 线上（高位在前），然后拉高 SCL，从机将在 SCL 高电平期间读取数据位，所以 SCL 高电平期间读取数据位，所以 SCL 高电平期间 SDA 不允许有数据变化，依次循环上述过程 8 次，即可发送一个字节。

先从高位起取出数据，再通过 SCL 拉高后拉低进行数据读取。

接收一个字节：

SCL 低电平期间，从机将数据位依次放到 SDA 线上（高位在前），然后拉高 SCL，主机将在 SCL 高电平期间读取数据位，所以 SCL 高电平期间 SDA 不允许有数据变化，依次循环上述过程 8 次，即可接收一个字节（主机在接收之前，需要释放 SDA，SDA 置 1）。




```
unsigned char I2C_ReceiveByte(void)
{
    unsigned char i,Byte=0x00;
    I2C_SDA=1;
    for(i=0;i<8;i++)
    {
        I2C_SCL=1;
        if(I2C_SDA){Byte|=(0x80>>i);}
        I2C_SCL=0;
    }
    return Byte;
}
```

为释放 SDA 先将 SDA 置 1，之后再将 SCL 拉高进行根据 SDA 的值进行每位数据的读取。

发送应答：

```
void I2C_SendAck(unsigned char AckBit)
{
    I2C_SDA=AckBit;
    I2C_SCL=1;
    I2C_SCL=0;
}
```

在接收完一个字节后，主机在下一个时钟发送一位数据，数据 0 表示应答，数据 1 表示非应答。

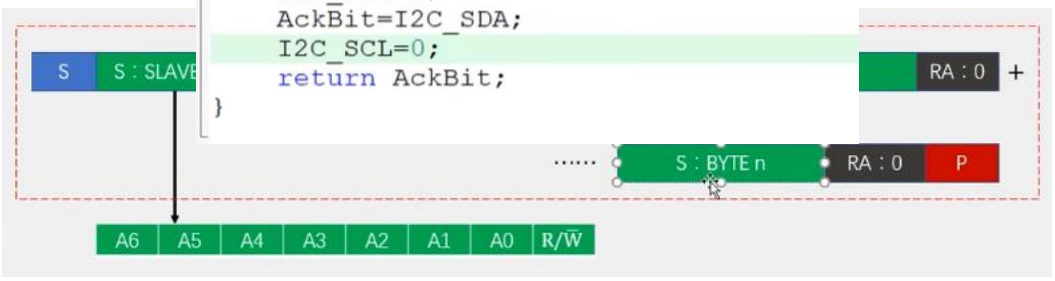
接收应答：在发送完一个字节后，主机在下一个时钟接收一位数据，判断从机是否应答，数据 0 表示应答，数据 1 表示非应答（主机在接收之前，需要释放 SDA）。

发送一帧数据：

格式：

起始条件 + 一个字节（从机地址+读写位） + 接收应答 + 字节 1 + 接收应答

```
unsigned char I2C_ReceiveAck(void)
{
    unsigned char AckBit;
    I2C_SDA=1;
    I2C_SCL=1;
    AckBit=I2C_SDA;
    I2C_SCL=0;
    return AckBit;
}
```



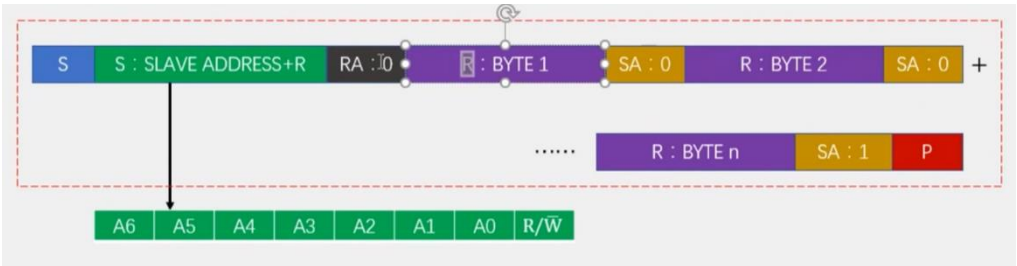
+ ……+ 字节 n + 接收应答 + 终止条件

程序地址：包括 7 位，A6~A0，其中 A6~A3 为固定位，A2~A0 可填入 I2C 地址。

读写位：0 为写，1 为读。

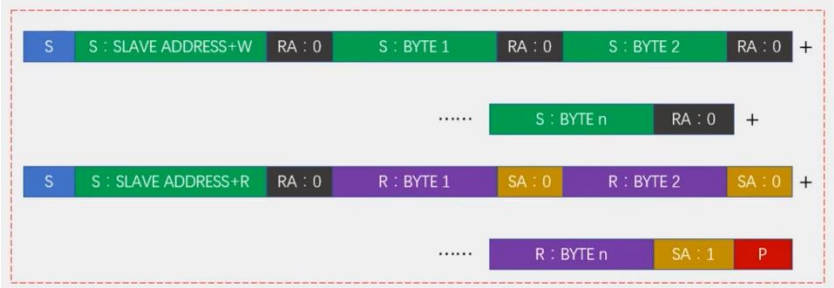
所完成的任务：向谁发什么。

接收一帧数据：



完成的任务：向谁收什么。

先发送再接收数据帧：



完成的任务：向谁收指定的什么。

AT24C02 **数据帧**

字节写：在 WORD ADDRESS 处写入数据 DATA

随机读：读出在 WORD DRESS 处的数据 DATA

AT24C02 的固定地址为 1010，可配置地址本开发板上为 000，所以 SLAVE ADDRESS+W 为 0xA0，SLAVE ADDRESS+R 为 0xA1