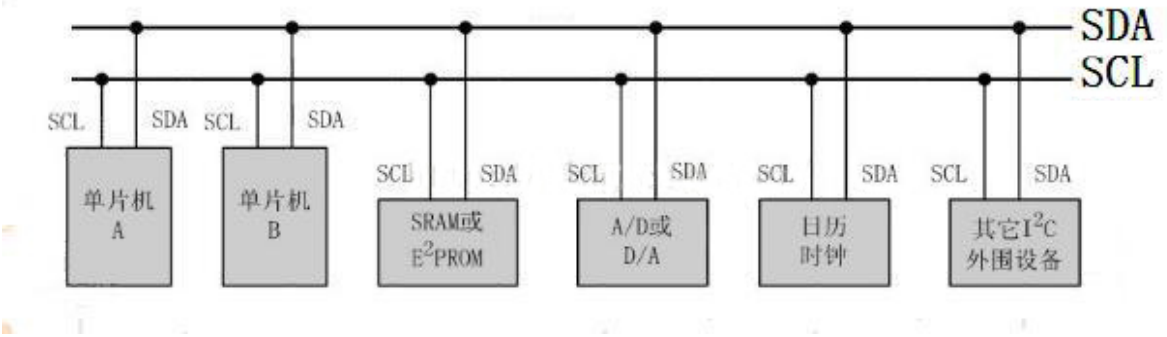
STM32第8次培训

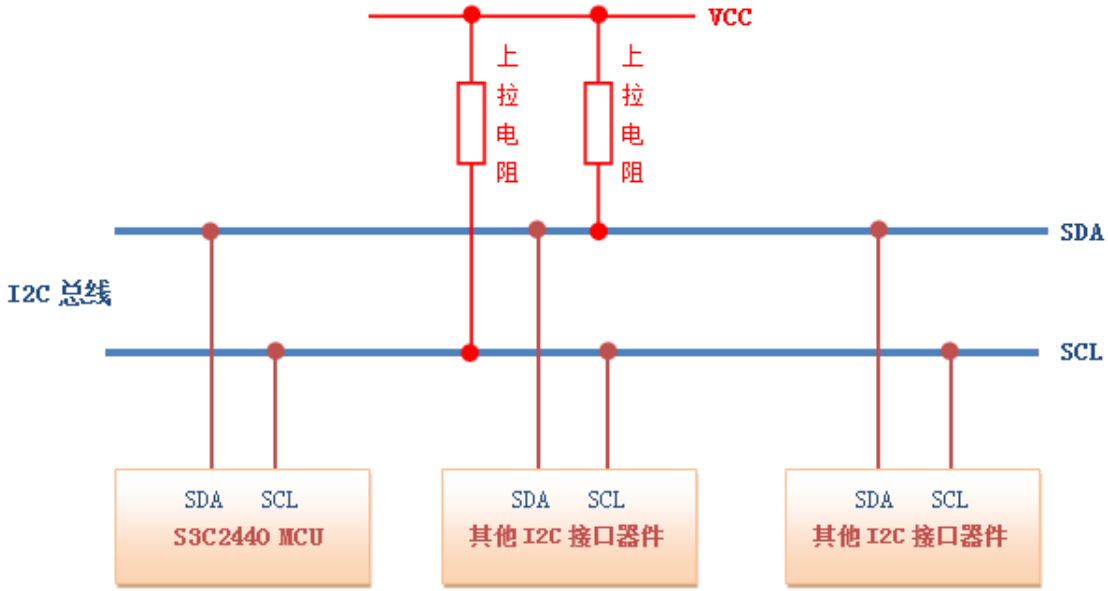
IIC简介

IIC (Inter - Integrated Circuit)内部集成总线，也可写作I2C，是PHILIPS 公司开发的**两线式串行总线**，用于多设备之间通讯，分为**主机Master**和**从机Slave**，**主机和从机可以有多个**，但**一般情况下只有一个主机**，**从机之间可以通过地址进行区分**，不同种类的设备地址不同，如果同时接入**多个相同种类的设备**，可以通过**片选信号**对从机进行选择。**通讯只能由主机发起**，支持的操作分为**读取和写入**，即主机读取从机的数据，以及向从机写入数据。

IIC串行总线一般有**两根信号线**，一根是**双向的数据线SDA**，另一根是**时钟线SCL**，其时钟信号是由主控器件产生。所有接到IIC总线设备上的**串行数据SDA都接到总线的SDA上**，各设备的**时钟线SCL接到总线的SCL上。**对于并联在一条总线上的每个**IC都有唯一的地址。**



一般情况下，**数据线SDA和时钟线SCL都是处于上拉电阻状态。**因为：在**总线空闲状态**时，这两根线一般被上面所接的**上拉电阻拉高，保持着高电平**



I2C 总线物理拓扑图

IIC特点

- **它是半双工通信方式**，同一时间只可以单向通信
- **它是同步通信**
- 支持不同速率的通讯速度，**标准速度(最高速度100kbps)，快速（最高400kbps)**

- 为了**避免总线信号的混乱**，IIC的**空闲状态**只能有**外部上拉**， 而此时空闲设备被拉到了高阻态，也就是相当于断路， 整个IIC总线只有开启了的设备才会正常进行通信，而不会干扰到其他设备。

IIC协议

IIC总线在传送数据过程中共有三种类型信号， 它们分别是：开始信号、结束信号和应答信号。

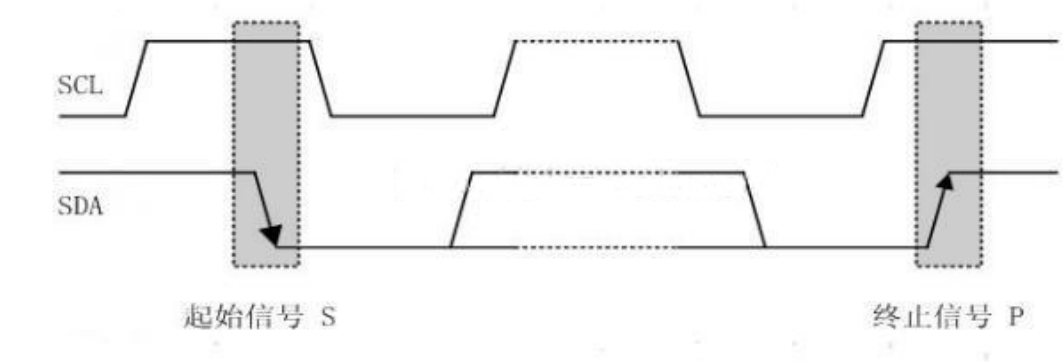
- **开始信号**：**SCL** 为**高电平**时，**SDA** 由**高电平向低电平跳变**，开始传送数据。
- **结束信号**：**SCL** 为**高电平**时，**SDA** 由**低电平向高电平跳变**，结束传送数据。
- **应答信号**：**主机SCL拉高，读取从机SDA的电平，为低电平表示产生应答**

  接收数据的 IC 在**接收到 8bit 数据后**，**向发送数据的 IC 发出特定的低电平脉冲**，表示已收到数据。**CPU** 向受控单元发出一个信号后，**等待受控单元发出一个应答信号**，CPU 接收到应答信号后，根据实际情况作出是否继续传递信号的判断。**若未收到应答信号，由判断为受控单元出现故障。**

IIC时序图

起始与停止信号

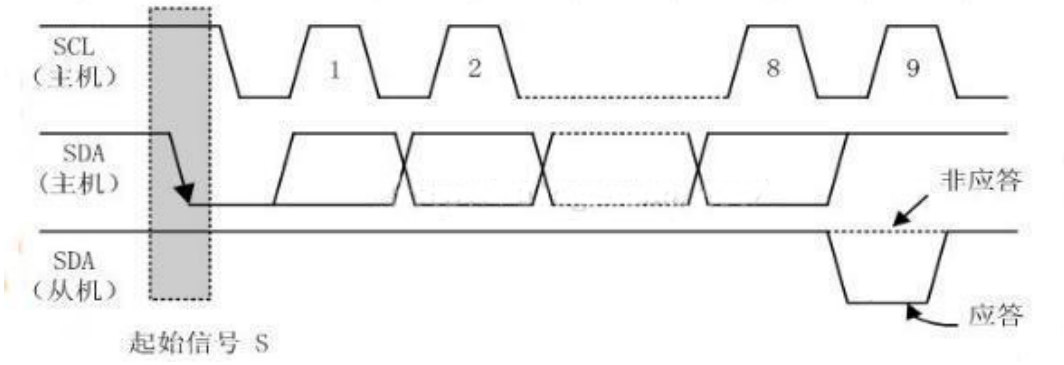
- **起始**信号：当时钟线**SCL为高期间**，数据线**SDA由高到低的跳变**；启动信号是一种电平跳变时序信号，而不是一个电 平信号；
- **停止**信号：当时钟线**SCL为高期间**，数据线**SDA由低到高的跳变**；停止信号也是一种电平跳变时序信号，而不是一个电平信号。



应答信号

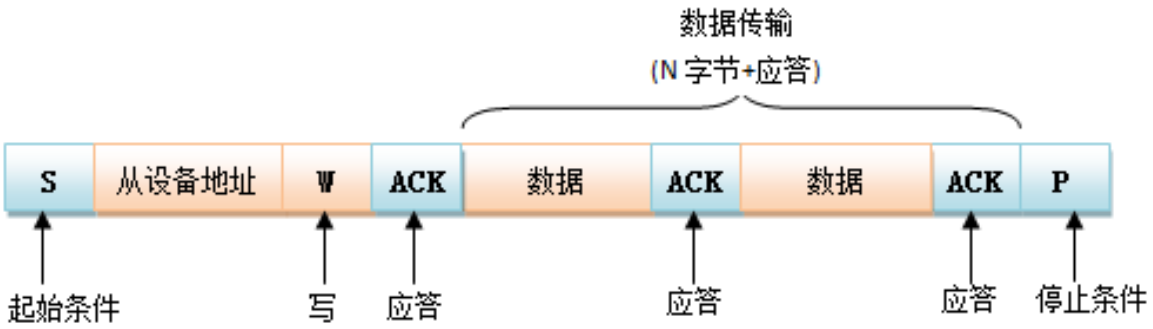
发送器每发送一个字节（8个bit），就在时钟脉冲9期间释放数据线，由接收器反馈一个应答信号。

- 应答信号为**低**电平时，规定为**有效应答位**（ACK，简称应答位），表示接收器已经成功地接收了该字节；
- 应答信号为**高**电平时，规定为**非应答位**（NACK），一般表示接收器接收该字节没有成功。

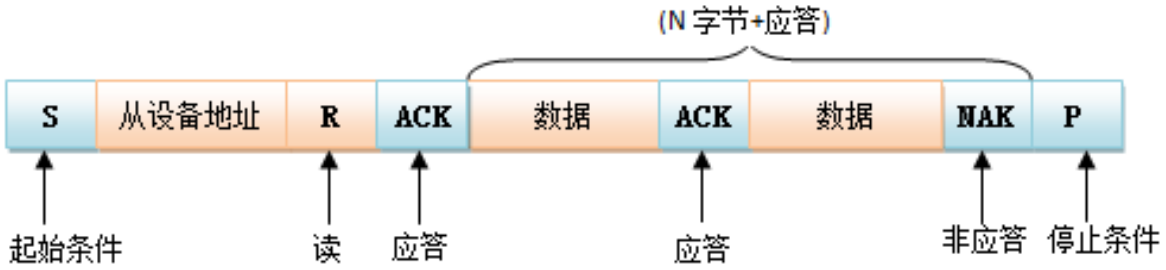


IIC数据传输

- 主设备在传输有效数据之前要**先指定从设备的地址**，地址指定的过程和上面数据传输的过程一样，只不过大多数从**设备的地址是7位**的，然后协议规定再给地址添加一个**最低位**用来**表示**接下来**数据传输的方向**，**0**表示主设备向从设备**写数据**，**1**表示主设备向从设备**读数据**。
- **主设备往从设备中写数据：**
- 此时的应答信号全都由从设备产生



- **主设备从从设备中读数据：**
- 在从机产生响应时，主机从发送变成接收，从机从接收变成发送。之后，数据由从机发送，主机接收，**每个应答由主机产生**，**时钟信号仍由主机产生。若主机要终止本次传输**，则发送一个**非应答信号**，接着主机产生停止条件。



代码部分

- main.c

```c
#include "main.h"

int main(void)
{
  HAL_Init();         //初始化HAL库
  Stm32_Clock_Init();     //初始化系统时钟
  led_init();         //初始化延时函数
  key_init();         //初始化LED
  delay_init();       //延时初始化
  OLED_Init();        //oled初始化
  OLED_Clear();       //清屏
  AT24CXX_Init();       //初始化at24c02

  OLED_ShowString(0,0,(uint8_t *)"test start");
  while(1)
  {
    key_control(0);
  }
}
```

- main.h

```
#ifndef __MAIN_H_
#define __MAIN_H_

#include "stm32f1xx_hal.h"
#include "key.h"
#include "led.h"
#include "sys.h"
#include "delay.h"
#include "stdio.h"
#include "oled.h"
#include "iic.h"
#include "at24cxx.h"

#endif
```

- iic.c

```
#include "iic.h"

void IIC_init(void)
{
  GPIO_InitTypeDef  gpio_handl;

  __HAL_RCC_GPIOB_CLK_ENABLE();

  //GPIOB8,B9初始化设置
  gpio_handl.Pin = GPIO_PIN_8 | GPIO_PIN_9 ;
  gpio_handl.Mode = GPIO_MODE_OUTPUT_PP;//推挽输出模式
  gpio_handl.Speed = GPIO_SPEED_HIGH;//高速
  gpio_handl.Pull = GPIO_PULLUP;//上拉

  HAL_GPIO_Init(GPIOB,&gpio_handl);//初始化

  IIC_SCL = 1; //拉高时钟线程
  IIC_SDA = 1; //拉高数据线程
}

/*发送IIC开始信号*/
void IIC_Start(void)
{
  SDA_OUT();//输出模式
  IIC_SCL = 1; //拉高
  IIC_SDA = 1; //拉高
  delay_us(4); //延时4us
  IIC_SDA = 0; //拉低数据线(开始:SCL为高，SDA由高->低时)
  delay_us(4);
  IIC_SCL = 0; //拉低SCL准备传输或接受数据
  delay_us(4);
}

/*发送IIC停止信号*/
void IIC_Stop(void)
{
  SDA_OUT();//输出模式
  IIC_SCL = 0; //保证SCL为低
  IIC_SDA = 0; //保证SDA为低
```

```
  delay_us(4);
  IIC_SCL = 1; //拉高时间线
  delay_us(4);
  IIC_SDA = 1; //(停止:SCL为高，SDA由低->高)
  delay_us(4);
}

//等待应答信号到来
//返回值：1，接收应答失败
//       0，接收应答成功
uint8_t IIC_Wait_Ack(void)
{
  uint8_t ucErrTime=0;
  SDA_IN();      //SDA设置为输入
  IIC_SDA=1;
  delay_us(1);
  IIC_SCL=1;
  delay_us(1);
  while(INPUT_SDA)
  {
    ucErrTime++;
    if(ucErrTime > 250)
    {
      IIC_Stop();
      return 1;
    }
  }
  IIC_SCL=0;//时钟输出0
  return 0;
}

/*产生应答*/
/*SCL:低->高->低*/
/*SDA:一直为低*/
void IIC_Ack(void)
{
  IIC_SCL = 0;
  SDA_OUT();
  IIC_SDA = 0;
  delay_us(2);
  IIC_SCL = 1;
  delay_us(2);
  IIC_SCL = 0;
}

/*不产生应答*/
/*SCL:低->高->低*/
/*SDA:一直为高*/
void IIC_Nack(void)
{
  IIC_SCL = 0;
  SDA_OUT();
  IIC_SDA = 1;
  delay_us(2);
  IIC_SCL = 1;
  delay_us(2);
  IIC_SCL = 0;
}
```

```c
/*
  主机写入数据到外设中
  参数值：data  要写入的一个字节
  返回值：NULL
*/
void IIC_Write_data(uint8_t data)
{
  uint8_t i;
  SDA_OUT(); //输出模式
  delay_us(4);
  IIC_SCL = 0;//SCL为低时才能写入数据
  for(i = 0;i < 8;i++)
  {
    IIC_SDA = (data&0x80)>>7;//高位先入(8位依次输入)
    data<<=1;//每次使data左移一位便可以传完一个字节
    delay_us(2);
    IIC_SCL = 1;//拉高及停止传输
    delay_us(2);
    IIC_SCL = 0;//拉低开启下次传输
    delay_us(2);
  }
}

/*
  主机从外设中读取一个字节的数据
  参数值：Ack(有无应答)
  返回值：recive
*/
uint8_t IIC_Read_Byte(uint16_t Ack)
{
  uint8_t i,receive = 0;
  SDA_IN();//SDA设置为输入
  for(i = 0;i < 8;i++)
  {
    IIC_SCL = 0;
    delay_us(2);
    IIC_SCL = 1;
    receive<<= 1;
    if(INPUT_SDA)
      receive++;
    delay_us(1);
  }
  if (!Ack)
    IIC_Nack();//无应答
  else
    IIC_Ack(); //应答

  return receive;
}
```

- iic.h

```
#ifndef __IIC_H_
#define __IIC_H_

#include "main.h"

//IO方向设置
/*SDA :表示数据线*/
/*SCL :表示时钟线*/
/*PB9*/
#define SDA_IN()    {GPIOB->CRH&=0XFFFFFF0F;GPIOB->CRH|=8<<4;} //上拉输入
#define SDA_OUT()   {GPIOB->CRH&=0XFFFFFF0F;GPIOB->CRH|=3<<4;} //推挽输出
//IO操作函数
#define IIC_SCL    PBout(8) //SCL
#define IIC_SDA    PBout(9) //输出SDA
#define INPUT_SDA  PBin(9)  //输入SDA

void IIC_init(void);                //初始化IIC的IO口
void IIC_Start(void);               //发送IIC开始信号
void IIC_Stop(void);                //发送IIC停止信号
void IIC_Write_data(uint8_t data);   //IIC写入一个字节
uint8_t IIC_Read_Byte(uint16_t ack); //IIC读取一个字节
uint8_t IIC_Wait_Ack(void);         //IIC等待ACK信号
void IIC_Ack(void);                  //IIC发送ACK信号
void IIC_Nack(void);                //IIC不发送ACK信号

void IIC_Write_One_Byte(uint8_t daddr,uint8_t addr,uint8_t data);

#endif
```

- key.c

```
#include "key.h"

void key_init()
{
  GPIO_InitTypeDef key_handl;

  key_handl.Mode = GPIO_MODE_INPUT;
  key_handl.Pin = GPIO_PIN_4 | GPIO_PIN_5;
  key_handl.Pull = GPIO_PULLUP;
  key_handl.Speed = GPIO_SPEED_FREQ_LOW;

  HAL_GPIO_Init(GPIOA,&key_handl);
}

//返回值为enum形式
KEY_STATUS key_scan(uint8_t mode)
{
  static uint8_t key_up = 1;
  if(mode == 1) key_up = 1;
  if(key_up && (KEY1 == 0 || KEY2 == 0))
  {
    key_up = 0;
    if(KEY1 == 0) return KEY1_PRESS;
```

```
    else if(KEY2 == 0) return KEY2_PRESS;
  }
  else if(KEY1 == 1 && KEY2 == 1)
    key_up = 1;
  return KEY_NULL;
}

/*只有功能函数发生了变化*/
void key_control(uint8_t mode)
{
  const char text_buffer[] = "AT24C02_TEST";//更换这里的字符串就可以更改写入的数据
  #define text_size sizeof(text_buffer)      //字符串的大小
  #define address 0                         //写入的地址
  uint8_t data_temp[text_size];              //存储读取的数据

  switch(key_scan(mode))
  {
    case KEY_NULL : break;
    case KEY1_PRESS :
    {
      AT24CXX_Write(address,(uint8_t *)text_buffer,text_size);//写入数据
      OLED_ShowString(0,0,(uint8_t *)"writting...");
      delay_ms(800);
       OLED_Clear();
      delay_ms(50);
      OLED_ShowString(0,0,(uint8_t *)"write is ok");
      break;
    }
    case KEY2_PRESS :
    {
      AT24CXX_Read(address,(uint8_t *)data_temp,text_size);
      OLED_ShowString(0,0,(uint8_t *)"reading...");
      delay_ms(800);
      OLED_Clear();
      delay_ms(50);
      OLED_ShowString(0,0,(uint8_t *)"read is ok");
      delay_ms(800);
      OLED_Clear();
      OLED_ShowString(0,2,(uint8_t *)data_temp);//显示读取的数据
      break;
    }
    default :break;
  }
}
```

- key.h （这里和之前的key.h相同）

```
#ifndef __KEY_H_
#define __KEY_H_

#include "main.h"

#define KEY1 HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_4)
#define KEY2 HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_5)
```

```c
/*用枚举形式定义*/
typedef enum
{
    KEY_NULL = 0,
    KEY1_PRESS,
    KEY2_PRESS,
}KEY_STATUS;

void key_init(void);
KEY_STATUS key_scan(uint8_t mode);
void key_control(uint8_t mode);

#endif
```