I2C 协议介绍及 STM32 实现

AI 航 团队

I2C（Inter-Integrated Circuit）通讯协议是由 Phiilps 公司开发的两线式串行总线，用于连接微控制器及其外围设备。是微电子通信控制领域广泛采用的一种总线标准，轻舟机器人驱动板上的惯导模块就是采用这种协议进行数据传输。

1. I2C 协议简介

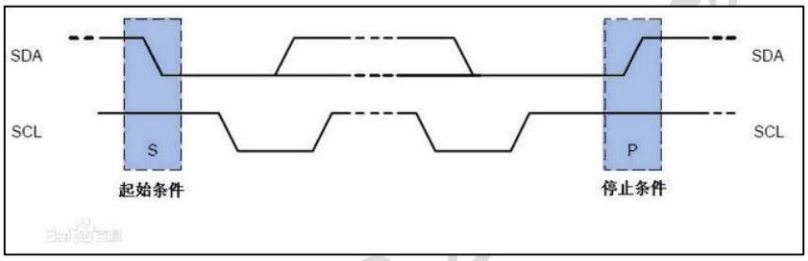
I2C 总线由两根线组成，分别是数据线 SDA 和时钟线 SCL，两条信号线同时处于高电平时，规定为总线的空闲状态。此时各个器件的输出级场效应管均处在截止状态，即释放总线，由两条信号线各自的上拉电阻把电平拉高。

I2C 通讯总线支持连接多个 I2C 设备，支持多个主机及多个通讯从机。为了区分设备，每个连接到总线的设备都有一个独立的地址，主机可以利用这个地址在不同设备之间进行通讯。本文重点介绍一个主机和一个从机的的情况，多机通信不作过多介绍。

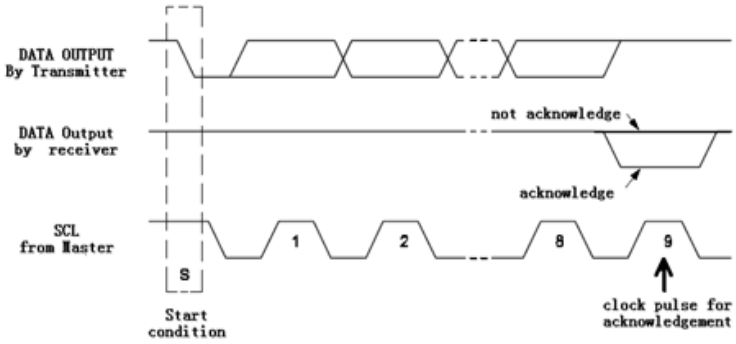
起始位与停止位的定义：

起始信号：当 SCL 为高期间，SDA 由高到低的跳变，表示通讯开始；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 SCL 为高期间，SDA 由低到高的跳变，表示通讯停止；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。

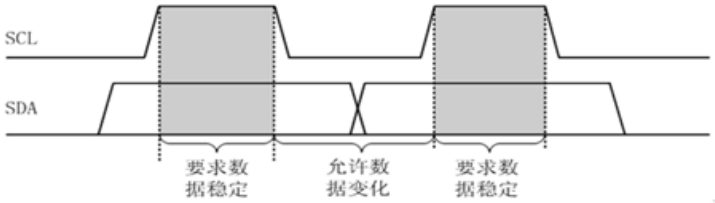


发送器每发送一个字节，就在时钟脉冲 9 期间释放数据线，由接收器反馈一个应答信号。应答信号为低电平时，规定为有效应答位（ACK 简称应答位），表示接收器已经成功地接收了该字节；应答信号为高电平时，规定为非应答位（NACK），一般表示接收器接收该字节没有成功。对于反馈有效应答位 ACK 的要求是，接收器在第 9 个时钟脉冲之前的低电平期间将 SDA 线拉低，并且确保在该时钟的高电平期间为稳定的低电平。如果接收器是主控器，则在它收到最后一个字节后，发送一个 NACK 信号，以通知被控发送器结束数据发送，并释放 SDA 线，以便主控接收器发送一个停止信号 P。



I²C 总线的响应

I2C 总线进行数据传送时，时钟信号为高电平期间，数据线上的数据必须保持稳定，只有在时钟线上的信号为低电平期间，数据线上的高电平或低电平状态才允许变化，也就是数据在 SCL 的上升沿到来之前就需准备好。



在 I2C 总线上传送的每一位数据都有一个时钟脉冲相对应（或同步控制），即在 SCL 串行时钟的配合下，在 SDA 上逐位地串行传送每一位数据。数据位的传输是边沿触发。

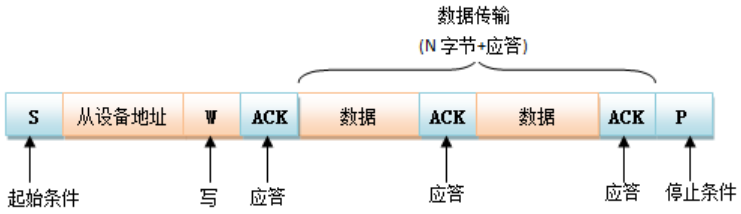
2. I2C 数据传送方式

I2C 在 stm32 中的实现方式有两种，一种是“硬件 I2C”，我们可以像开发 UART 串口那样，通过对 stm32 的片上外设进行配置，外设对应的引脚就会自动按照协议产生通讯信号，进行数据收发，这种实现方式起来比较简单。另外一种是用两个 IO 引脚模拟 SDA 和 SCL 电平，从而实现通讯，这种方式被称为“软件 I2C”。

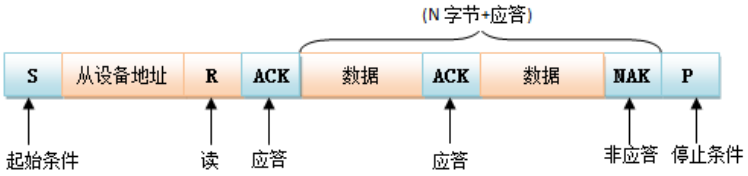
为了使大家更好的理解 I2C 协议，我们接下来将学习第二种方式，这种方式用途很广，即使换做任何一种控制芯片，我们都可以通过器 IO 来模拟 I2C 协议，轻舟机器人对 IMU 的数据读取也是用的这种方式。

对 I2C 总线的操作实际就是主从设备之间的读写操作。大致可分为以下三种操作情况：

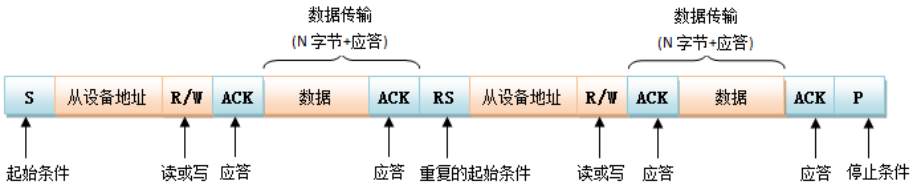
第一，主设备往从设备中写数据。数据传输格式如下：



第二，主设备从从设备中读数据。数据传输格式如下：



第三，主设备往从设备中写数据，然后重启起始条件，紧接着从从设备中读取数据；或者是主设备从设备中读数据，然后重启起始条件，紧接着主设备往从设备中写数据。数据传输格式如下：



第三种操作在单个主设备系统中，重复的开启起始条件机制要比用 STOP 终止传输后又再次开启总线更有效率。

3. Stm32 编程实现

下面将结合轻舟机器人驱动板来讲述软件模拟 I2C 通讯编程流程。

Myiic.h 文件内容---相关宏定义

```
#ifndef __MYIIC_H
#define __MYIIC_H
#include "sys.h"

// IO 方向设置
#define SDA_IN()  {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=8<<12;}
#define SDA_OUT() {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=3<<12;}

// IO 操作函数
#define IIC_SCL    PBout(10)          //SCL
#define IIC_SDA    PBout(11)          //SDA
#define READ_SDA   PBin(11)           //输入 SDA

// IIC 所有操作函数
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_Send_Byte(u8 txd);           //IIC 发送一个字节
u8 IIC_Read_Byte(unsigned char ack);  //IIC 读取一个字节
u8 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(u8 daddr,u8 addr,u8 data);//写一个字节
u8 IIC_Read_One_Byte(u8 daddr,u8 addr);          //读一个字节
#endif
```

Myiic.c 文件内容---相关宏定义

初始化函数

```
#include "myiic.h"
#include "delay.h"

void IIC_Init(void)     //初始化 IIC
{
   GPIO_InitTypeDef GPIO_InitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);   //使能 PB 端口时钟
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10|GPIO_Pin_11;      //端口配置
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;          //推挽输出
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;          //2M
   GPIO_Init(GPIOB, &GPIO_InitStructure);                    //根据设定参数初始化 GPIOB

       IIC_SCL=1;
       IIC_SDA=1;
}
```

产生起始信号和停止信号

```
//产生 IIC 起始信号，在 SCL 高电平时，SDA 出现一个下调沿表示 I2C 启动信号
void IIC_Start(void)
{
	SDA_OUT();	//sda 线输出
	IIC_SDA=1;
	IIC_SCL=1;	//产生一个时钟
	delay_us(4);
	IIC_SDA=0;	//当 SCL 在好电平期间 SDA 拉低
	delay_us(4);
	IIC_SCL=0;	//钳住 I2C 总线，准备发送或接收数据
}
//产生 IIC 停止信号
void IIC_Stop(void)
{
	SDA_OUT();	//sda 线输出
	IIC_SCL=0;
	IIC_SDA=0;
	delay_us(4);
	IIC_SCL=1;	//产生一个时钟
	delay_us(4);
	IIC_SDA=1;	//发送 I2C 总线结束信号
}
```

等待应答信号到来函数

```
//返回值：1，接收应答失败
//		0，接收应答成功
u8 IIC_Wait_Ack(void)
{
	u8 ucErrTime=0;
	SDA_IN();	//SDA 设置为输入
	IIC_SDA=1;delay_us(1);
	IIC_SCL=1;delay_us(1);
	while(READ_SDA)
	{
		ucErrTime++;
		if(ucErrTime>250)
		{
			IIC_Stop();
			return 1;
		}
	}
	IIC_SCL=0;	//时钟输出 0
	return 0;
}
```

应答相关函数

```
//产生 ACK 应答
void IIC_Ack(void)
{
	IIC_SCL=0;	//时钟信号低电平
	SDA_OUT(); //输出
	IIC_SDA=0;	//数据信号低电平
	delay_us(2);
	IIC_SCL=1;	//时钟信号高电平
	delay_us(2);
	IIC_SCL=0;	//时钟信号低电平
}
//不产生 ACK 应答
void IIC_NAck(void)
{
	IIC_SCL=0;	//时钟信号低电平
	SDA_OUT(); //输出
	IIC_SDA=1;	//数据信号高电平
	delay_us(2);
	IIC_SCL=1;	//时钟信号高电平
	delay_us(2);
	IIC_SCL=0;	//时钟信号低电平
}
```

发送一个字节

```
//返回从机有无应答
//1，有应答
//0，无应答
void IIC_Send_Byte(u8 txd)
{
    u8 t;
    SDA_OUT();
    IIC_SCL=0;                    //拉低时钟开始数据传输
    for(t=0;t<8;t++)
    {
        IIC_SDA=(txd&0x80)>>7;// 取字节的最高位的值，将其右移 7 位
        txd<<=1;                  //然后将 txd 数据左移一位，等待下一个循环时发送
        delay_us(2);              //对 TEA5767 这三个延时都是必须的
        IIC_SCL=1;                //产生一个时钟
        delay_us(2);
        IIC_SCL=0;
        delay_us(2);
    }
}
```

读取一个字节

```
//读 1 个字节，ack=1 时，发送 ACK，ack=0，发送 nACK
u8 IIC_Read_Byte(unsigned char ack)
{
    unsigned char 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;   //产生一个时钟
        receive<<=1;//第一个循环左移不受影响，因为为 0，之后每循环一次左移一位
        if(READ_SDA) receive++; //READ_SDA 为读取输入脚的电平，为 1 的话将最低位置 1
        delay_us(1);
    }
    if (!ack)
        IIC_NAck();   //发送 nACK
    else
        IIC_Ack();    //发送 ACK
    return receive;
}
```

I2C 的协议以及软件实现协议过程就讲述到这里，具体应用方法将在下一节 IMU9250 数据读取中讲述，请大家继续学习，本节内容网上类似教程较多，可以对比着理解。