

# CAFS4000



无锡康森斯克电子科技有限公司

CAFS 气体质量流量计 I2C

通信协议

V1.2V



## 1.适用范围

本协议适用于具有 I2C 通讯接口的 CAFS4000 气体质量流量计，其软件版本为 V1.2V。

## 2.CAFS3000 I2C 总线协议

### 2.1 I2C 协议简介

I2C 总线 (I2C bus, Inter-IC bus) 是一个双向的两线连续总线，提供集成电路 (ICs) 之间的通信线路。I2C 总线是一种串行扩展技术，最早由 Philips 公司推出，广泛应用于电视，录像机和音频设备。I2C 总线的意思是“完成集成电路或功能单元之间信息交换的规范或协议”。Philips 公司推出的 I2C 总线采用一条数据线 (SDA)，加一条时钟线 (SCL) 来完成数据的传输及外围器件的扩展；对各个节点的寻址是软寻址方式，节省了片选线，标准的寻址字节 SLAM 为 7 位，可以寻址 127 个单元。

I2C 总线有三种数据传输速度：标准，快速模式和高速模式。标准的是 100Kbps，快速模式为 400Kbps，高速模式支持快至 3.4Mbps 的速度。所有的与次之传输速度的模式都是兼容的。I2C 总线支持 7 位和 10 位地址空间设备和在不同电压下运行的设备。

### 2.2 CAFS3000 I2C 功能及内容

CAFS3000 的 I2C 工作协议具有 2 个功能：

- 1.对 I2C 地址进行修改，实现多机通信功能
- 2.读取当前流量值

## 2.2.1 修改 I2C 地址操作内容

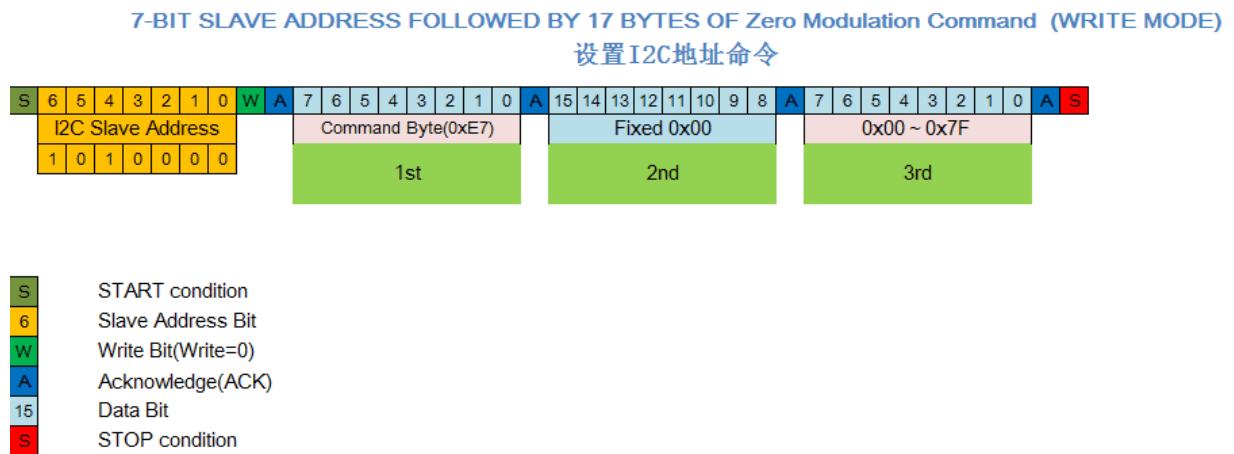


图 2.1 修改 I2C 地址通信格式

通过对指定地址进行 I2C 写操作，可以对 I2C 地址（即 I2C Slave Address）进行修改，CAFS3000 出厂默认 I2C 地址为 0x50,修改后，地址更改为所修改的值。修改 I2C 地址写操作命令有 3 个字节数据，第一个字节为寄存器地址，数据为 0xE7，此数据不可更改，否则操作无效。第二个字节为所变更的 I2C 地址高字节，数据为 0x00，因为 I2C 总线最多可以串联 127 个器件，因此 I2C 地址高字节必须为 0x00，否则会导致地址错误。第三个字节为所变更的 I2C 地址低字节，范围为 0x00~0x7F,可根据需要自行选择地址数据。

在发送 I2C 地址修改写操作命令后，I2C 地址即改变为所变更的地址，用户可以对流量进行读操作来确认是否修改成功，若读取到流量值，则说明修改成功，反之说明失败，需重新操作。

## 2.2.2 读取当前流量协议内容

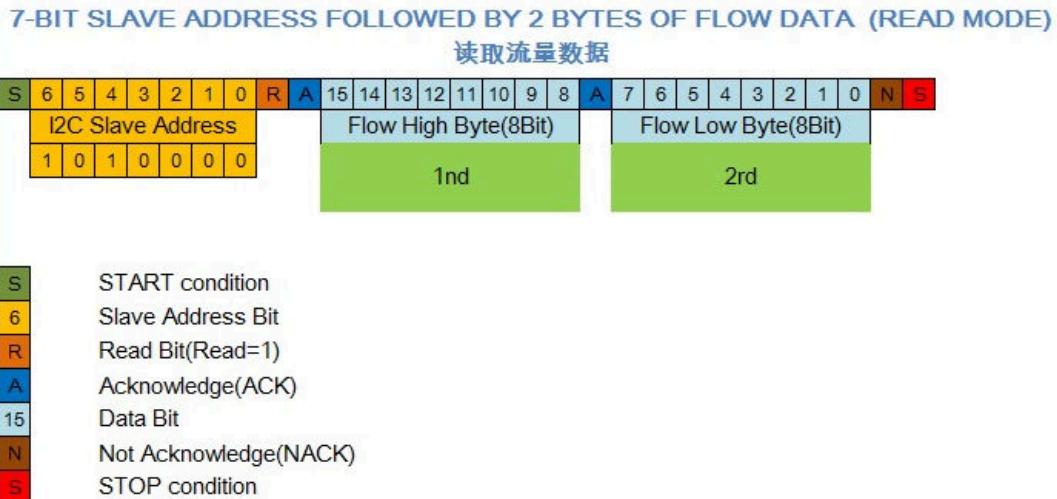


图 2.2 读取流量数据通信格式

通过对指定地址读取数据，可以得到当前流量值。流量数据分为 2 个字节，第一个字节为流量数据高字节，第二个字节为流量数据低字节，流量数据是实际流量的 100 倍，因此实际流量精确到小数点后两位。流量数据默认单位是 L/M (升/分钟)，例如读取到 16 进制数据为 03 E8，则转换为 10 进制当前实际流量为 10.00L/M，若需要流量单位为 M3/H (立方米/小时)，则通过下面公式可以计算得出(M 为立方米/小时流量，L 为升/分钟流量)：

$$M = L/1000 * 60$$

计算得出当前流量为 0.6M3/H。

## 例程一：流量传感器模块标准 DEMO 程序

```
//读写流量传感器模块例程
//SDA, SCL 分别对应单片机相应的 IO 口
//主机发 0XA1 给从机, 从机回 5 字节数据给主机

#include"IIC_Master.h"
#define SDA PA0
#define SCL PA1
Unsigned int IIC_RX_Buf[5],IIC_TX_Buf[5];
bit ErrorBit;
void I2C_Init(void)
{
    SDA_INPUT=0;                                //程序初始化将 SDA 引脚设置为输出
    SCL_INPUT=0;                                //程序初始化将 SCL 引脚设置为输出
    SDA=1;
    SCL=1;
}
Void I2C_Start(void)
{
    SDA=1;
    Delay_Us(1);

    SCL=1;
    Delay_Us(1);

    SDA=0;
    Delay_Us(1);

    SCL=0;
    Delay_Us(1);
}

void I2C_Stop(void)
{
    SCL=0;
    Delay_Us(1);

    SDA=0;
    Delay_Us(1);

    SCL=1;
    Delay_Us(1);

    SDA=1;
    Delay_Us(1);
```

```

}

//-----
void I2C_ACK(void)
{
    SDA=0;
    Delay_Us(1);

    SCL=1;
    Delay_Us(1);

    SCL=0;
    Delay_Us(1);

}

//-----
void I2C_NoAck(void)
{
    SDA=1;
    Delay_Us(1);

    SCL=1;
    Delay_Us(1);

    SCL=0;
    Delay_Us(1);

}

//-----
Unsigned int I2C_ReadByte(void)
{
    Unsigned int ucValue=0;
    Unsigned int ucIndex;

    SDA=1;
    Delay_Us(1);
    SDA_INPUT=1; //将 SDA 引脚设置为输入
    Delay_Us(1);
    for ( ucIndex = 0; ucIndex < 8; ucIndex++ )
    {
        ucValue <= 1;
        SCL=0;
        Delay_Us(1);
        SCL=1;
        Delay_Us(1);
        if(IIC_DAT==1) // IIC_DAT 就是将 SDA 设置为输入后，读出
SDA 引脚的电平值
        {
            ucValue = ucValue | 0x01;
        }
        else
    }
}

```

```

        { ucValue = ucValue & 0xfe;}
        Delay_Us(1);
        SCL=0;
        Delay_Us(1);
    }
    SDA_INPUT=0;           //将 SDA 设置为输出
    Delay_Us(1);
    return ucValue;
}

//-----
void    I2C_WriteByte( unsigned int ucData )
{
    u8 i;
    for( i = 0; i < 8; i++ )
    {
        SCL=0;
        Delay_Us(1);

        if((ucData & 0x80) == 0x80)
        {
            SDA=1;
            Delay_Us(1);
        }
        else
        {
            SDA=0;
            Delay_Us(1);
        }

        SCL=1;
        Delay_Us(1);

        SCL=0;
        Delay_Us(1);
        ucData <<= 1;
    }
    SCL=1;
    Delay_Us(1);
    ErrorBit = IIC_DAT;
    Delay_Us(1);
    SCL=0;
    Delay_Us(1);
}

void    iic_master_proc(void)
{

```

```
Unsigned int count=5,i;  
I2C_Init();  
I2C_Start();  
I2C_WriteByte(0xa1); //写地址 0 X A 1  
for(i = 0;i < count;i++)  
{  
    IIC_RX_Buf[i] = I2C_ReadByte();  
    if(i < (count -1))  
        I2C_ACK();  
    else  
        I2C_NoAck();  
}  
I2C_Stop();  
}
```