CRC 校验错误问题说明

芯片 CRC 校验时常见问题及解决办法如下。本文档适用芯片包括：

1、数字温度传感芯片系列：MY18E20、MY1820、MY18B20Z、MY18B20L、M601、M1601、M1820、M117、MTS01 等；

2、数字电容传感芯片系列：MDC02、MDC04 等。

一、 单总线通信

1.1 读出寄存器值与 CRC 校验值全为 0xFF

1. reset 后有 presence

原因 1：读时序不合适，从读 bit 函数第一个下降沿到读值的实际时间超过了 15us，错过了正确的采样时间。

解决方法：确认延时是否准确，确保从读 bit 函数第一个下降沿到读值的实际时间为 13us。

原因 2：GPIO 若为推挽模式，读取时 GPIO 输入模式切换失败。

解决方法：按照读取 presence 时的切换方式读取寄存器值。

2. reset 后无 presence

原因 1：DQ 与 VDD 短路。

解决方法：检查硬件，保证 DQ 与 VDD 间无短路现象。

原因 2：由于供电电压偏低、测量环境温度偏低或线缆过长，reset 时长不够导致没有响应。

解决方法：拉长 reset 时长。

原因 3：GPIO 若为推挽模式，读取时 GPIO 输入模式切换失败。

解决方法：检查输入模式下读取的寄存器是否为输入数据寄存器，切换输入模式后是否给够延时保证切换完成后再读取寄存器值等。

1.2 读出寄存器值与 CRC 校验值全为 0x00

原因 1：没接上拉电阻。

解决方法：在 VDD 和 DQ 间按需求接一个上拉电阻。

注：多点级联或长线缆等应用条件下上拉电阻优选 1KΩ。

原因 2：程序延时不准，导致程序中读取 presence 为 1 后 return FALSE，无法进行后面的测温及读取。

解决方法：调整延时，例程中的延时都是指实际时间，测出程序中 1us 实际是多长时间，凑出例程中的延时时间，或是用 Nop 构造出实际时间为 1us 的延时函数，按照例程中给的延时长度给出相同时长的延时。

1.3 读出寄存器值正确，只有 CRC 校验值错误

原因：读时序从读 bit 函数第一个下降沿到读值的实际时间稍长，错过 CRC 的正确采样时间。

解决方法：稍微缩短读时序从读 bit 函数第一个下降沿到读值的实际时间，使其能正确读到寄存器值和 CRC 校验值。

1.4 读出寄存器值错误但 CRC 校验值正确

原因：根据具体寄存器值判断值判断。

解决方法：请联系 *sales@mysentech.com* 解决。

二、 I2C 通信

2.1 读出寄存器值与 CRC 校验值全为 0xFF（地址位无响应）

原因 1：SDA 与 VDD 短路。

解决方法：检查硬件，保证 SDA 与 VDD 间无短路现象。

原因 2：唤醒失败（确认硬件连接无误后，逻辑分析仪/示波器可以抓取到芯片 SDA/SCL 上已收到正确的主机时序，在第一次发送从机地址后从机未在第 9bit 应答，但后续是有应答的，怀疑是芯片处于低功耗睡眠模式未被唤醒导致）。

解决方法：

- 若基于软件 IIC 构造时序，建议同时做到两点：

(1) 增大正常通信的 $t_{HD;STA}$ 参数，即从 Start 到 SCL 首次拉低的时间长度（定义可以参见产品手册中 I2C 总线时序参数图），在不同 VDD 电压下建议配置为不低于表 1 给出的数值：

VDD (V)	5	4	3	2	1.8
$t_{HD;STA\ MIN}$ (us)	30	35	50	60	150

表 1：建议配置 $t_{HD;STA}$ 的最小值

(2) 在正式发送 I2C 指令前，循环 5 次执行图 3 所示的 dummy 序列，具体代码见图 4。请注意，在 dummy 序列期间，SDA 首次拉低的时间长度需要依照表 1 来配置。

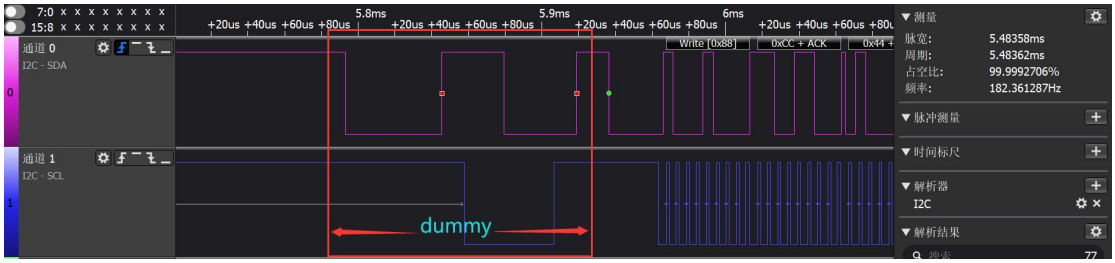

图 3：基于软件 I2C 的 dummy 序列波形图

```
void dummy_start(void)
{
    //---------2----------//
    i2c_set_scl();
    i2c_clear_sda();
    //---------3----------//
    Delay_us(150);
    //---------4----------//
    i2c_set_sda();
    Delay_us(10);
    i2c_clear_scl();
    Delay_us(20);
    //---------5----------//
    i2c_clear_sda();
    Delay_us(25);
    //---------6----------//
    i2c_set_scl();
    Delay_us(10);
    i2c_set_sda();
    //---------7----------//
    Delay_us(15);
    //---------8----------//
}
```

图 4：基于软件 I2C 的 dummy 序列代码

- 若基于硬件 IIC 通信，在无法更改 $t_{HD;STA}$ 参数的前提下，建议同时做到两点：

(1) 增大正常通信的 $t_{START-DELAY}$ 参数，即从 SCL 首次拉低到 SCL 首次拉高的延时长度，在不同 VDD 电压下建议配置为不低于表 2 给出的数值：

VDD (V)	5	4	3	2	1.8

tSTART-DELAY MIN (us)	30	35	50	60	150

表 2：建议配置 $t_{START-DELAY}$ 的最小值

(2) 在正式发送 I2C 指令前，循环 5 次执行图 5 所示的基于”Start+0x7E 数据（左移读写位后实际发送 0xFC）+Stop”的 dummy 序列，具体代码见图 6。

该序列仅用于唤醒从机，但数据不会被从机响应。请注意，在 dummy 序列期间，从 SCL 首次拉低到 SCL 首次拉高的延时长度需要依照表 2 来配置。

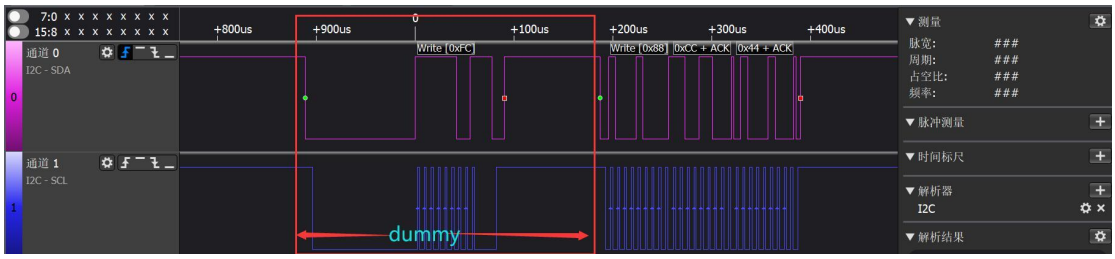

图 5：基于硬件 I2C 的 dummy 序列波形图

```
void dummy(void)
{
    int timeout;
    I2C_GenerateSTART(I2C2, ENABLE); //  发送起始位
    Delay_us(150);
    timeout=10000;
    while(!I2C_CheckEvent(I2C2,
    I2C_EVENT_MASTER_MODE_SELECT)) //EV5,主模式
    {
        timeout--;
        if(timeout==0)
        {break;}
    }
    I2C_Send7bitAddress(I2C2, 0xFC,
    I2C_Direction_Transmitter); //发送器件地址(写)
    Delay_us(200);
    I2C_GenerateSTOP(I2C2, ENABLE); //发送停止位
    Delay_us(200);
}
```

图 6：基于硬件 I2C 的 dummy 序列代码

其他说明：

(1) 上述 dummy 序列中的 delay_us 建议按照参考例程添加，若只能使用硬件

IIC 且并未配置其他延时函数的话可考虑省去。

(2) 无论是硬件 IIC 还是软件 IIC，我们都建议按照参考例程加入用于帮助从机唤醒的 dummy 序列，且最好执行至少 5 次再正式进行数据通信(Start-发送从机地址+ACK-发送指令+ACK….-Stop)。在添加 dummy 序列后，当发送从机地址可以收到应答，即可以进行后续正常读写操作。

2.2 读出寄存器值为 0xFF，CRC 校验值正确(地址位有响应)

原因 1：配置 GPIO 后未把 SDA 、SCL 置 1。

解决方法：配置 GPIO 后先把 SDA 、SCL 置 1。

原因 2：读取流程有问题，例如单字节读写，地址位和指令字节发完后发了 stop 再发读字头或写字头读写数据。

解决方法：按照手册中的读写流程进行操作。

2.3 读出寄存器值与 CRC 校验值全为 0x00

原因 1：未接上拉电阻。

解决方法：分别在 VDD 和 SDA、VDD 和 SCL 间按需求接一个上拉电阻。

原因 2：SDA 与 GND 短路或 SDA 断路。

解决方法：检查硬件，保证 SDA 与 GND 间无短路、或 SDA 无断路现象。