8051 系列

I2C应用笔记

Rev. 1.0.0

目录

1. 概述

1.1 目的

本文档介绍了 I2C 的功能及工作模式，阐述了 I2C 的应用注意使用。

1.2 内容提要

本文档包含以下内容：

第 2 章：功能概述。

第 3 章：功能特性。

第 4 章：功能描述。

第 5 章：工作模式。

第 6 章：应用注意事项。

2. 功能概述

I2C 是一种两线双向串行总线，为设备之间的数据交换提供了一种简单有效的连接方式。

I2C 是一个真正的多主机总线，包含了冲突检测和仲裁机制。冲突检测和仲裁机制用来在两个或多个主机同时尝试控制总线的情况下，防止数据损坏。

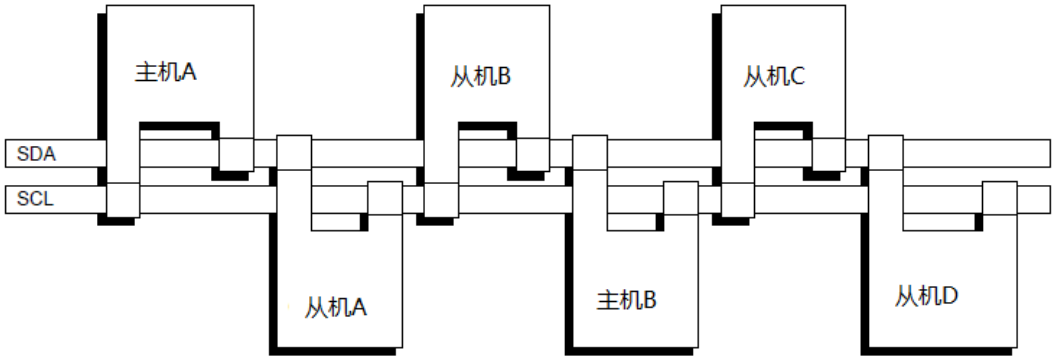

图 2-1：I2C 总线连接示意图

3. 功能特性

- 支持 4 种工作方式：主控发送、主控接收、从动发送、从动接收。
- 支持 2 种传输速度模式：
  - 标准（高达 100Kb/s）；
  - 快速（高达 400Kb/s）。
- 执行仲裁和时钟同步。
- 支持多主机系统。
- 主机方式支持 I2C 总线上的 7 位寻址模式。
- 从机方式支持 I2C 总线上的 7 位寻址模式。
- 中断功能。

4. 功能描述

4.1 数据有效性

SDA 线上的数据在 SCL 的高电平时间内必须保持稳定，SDA 的电平状态可在 SCL 的低电平时间内改变。每传输一位数据产生一个时钟。

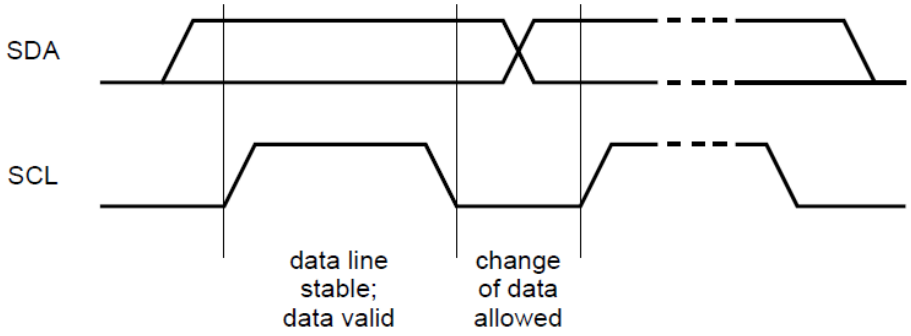

图 4-1： I2C 数据有效性

4.2 START 与 Repeated START

I2C 总线上数据的传输以 START（启动）信号开始，I2C 总线便进入忙碌状态。主机可以发送 Repeated START（重复启动信号）改变数据的发送方向（如：写入从机改变为读取从机），并且 I2C 总线继续保持忙碌状态。

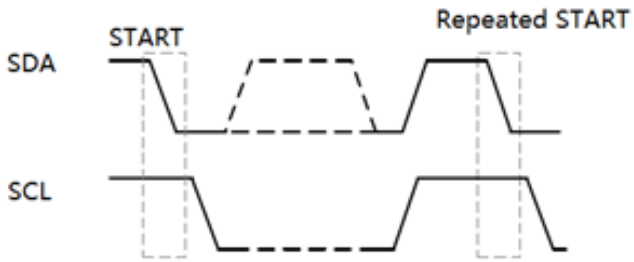

图 4-2：START 与 Repeated STATR

如图 4-2 所示，当 I2C 总线空闲时，SDA 与 SCL 在上拉电阻的作用下呈现出高电平。此时主机可发送 START 信号开始新的数据传输。

当 SCL 为高电平时，SDA 从高电平跳变到低电平代表 START（启动）信号。Repeated START（重复启动）信号与 START 信号产生条件相同。

4.3 STOP

当 SCL 为高电平时，SDA 从低电平跳变到高电平代表 STOP（停止）信号。意味数据传输的结束，I2C 总线进入空闲状态。

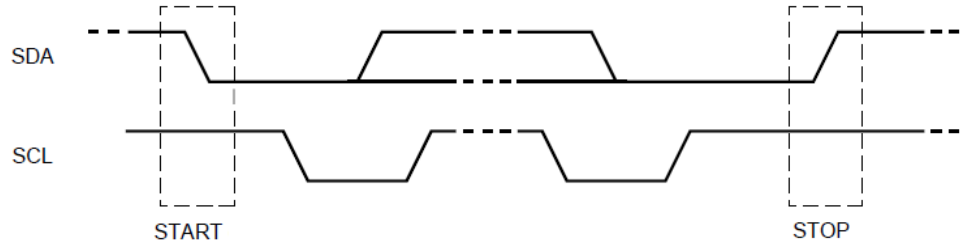

图 4-3：START 与 STOP

4.4 数据传输格式

I2C 总线上以 8 位长度的数据（1Byte）作为传输单位，每个 Byte 后紧随着 1 位应答位。数据以高位在前（MSB），低位在后（LSB）的顺序进行传输。

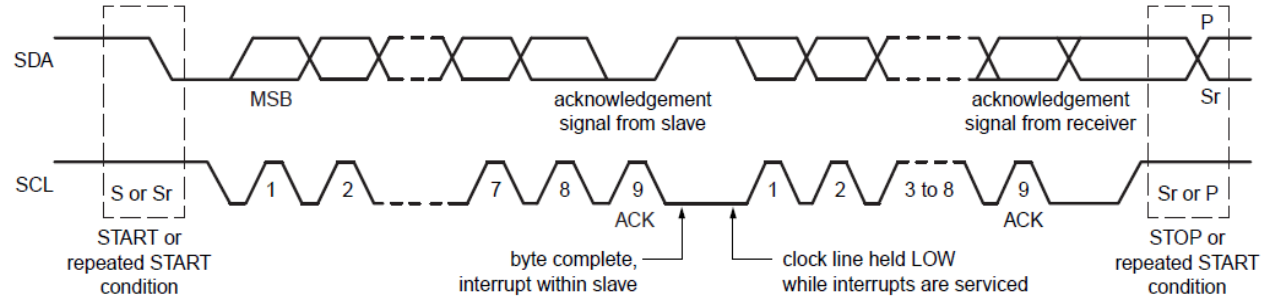

图 4-4：数据传输

4.5 ACK 与 NACK

每个 Byte 数据发送后都会紧跟着第 9 个 SCL 时钟信号传输 ACK（应答）信号。在第 9 个 SCL 高电平区间，SDA 为低电平代表有应答信号，如果 SDA 为高电平，代表无应答信号。

如图 4-4 所示，在主机寻址或者传输数据到从机时，主机可根据从机是否回复 ACK 判断从机的状态如果从机未回复 ACK 信号，主机可以发送 STOP 信号结束传输或者发送 Repeated START 信号开启新的数据传输。在主机接收数据时，不发送 ACK 信号给从机，从机将会释放 SDA 控制权，主机可发送 STOP 信号结束传输和 Repeated START 信号开启新的数据传输。

4.6 时钟同步

两个主机可以同时在一个 I2C 总线上传输，通过时钟同步和仲裁决定哪一个主机控制总线并完成其传输。在单个主机的系统中时不需要同步和仲裁。

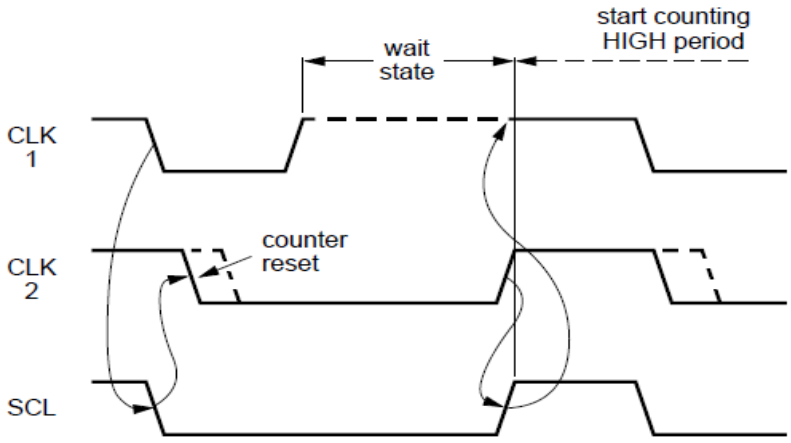

图 4-5：时钟同步

如图 4-5 所示，SCL 的电平是主机 1 与主机 2 的时钟信号相与的结果。

4.7 主机仲裁

仲裁功能保证了数据正确传输。

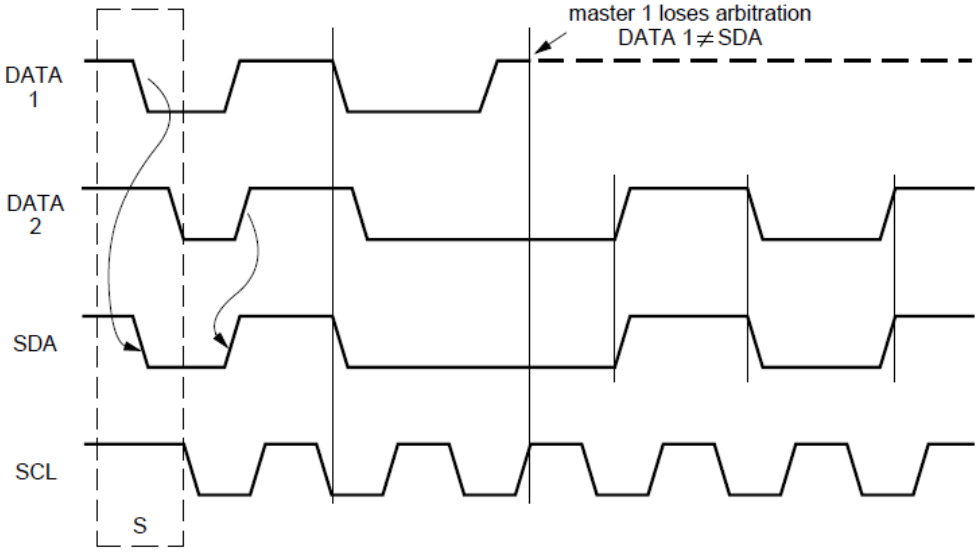

图 4-6：仲裁

如图 4-6 所示，SDA 的电平是主机 1 和主机 2 的 DATA 相与的结果。主机每次发送数据时都会检测数据是否正确发送。当主机检测到发送的数据与 SDA 上的数据不匹配时，主机就会丢失仲裁权。如图 4-6 中主机 1 发送的数据与 SDA 不匹配，主机 1 丢失仲裁权。当主机 1 在发送数据时丢失仲裁权，主机 1 在丢失仲裁后完成剩余时钟的发送后便进入从机模式。赢得仲裁权的主机 2 将会继续发送数据。

4.8 从机地址

I2C 模块支持 7 位地址模式。

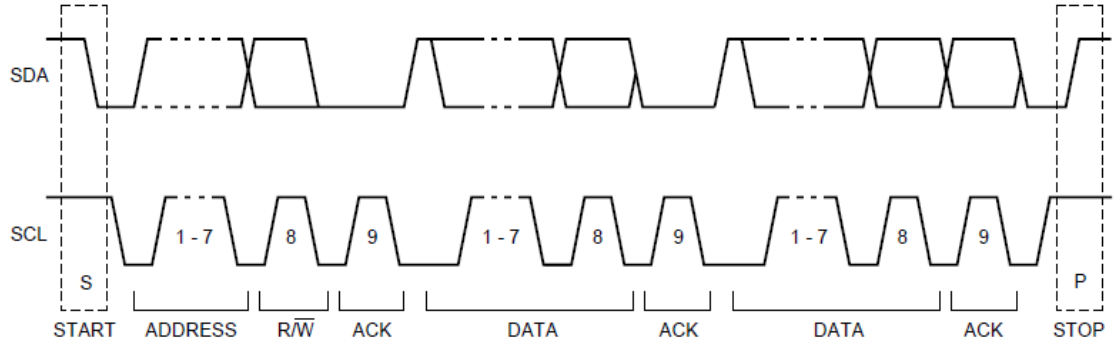

图 4-7：数据传输

4.8.1 7 位地址模式

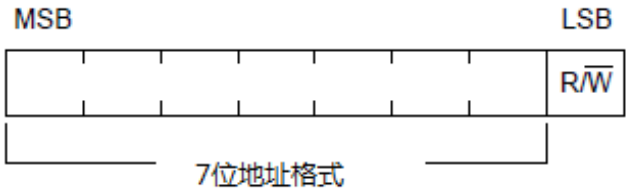

图 4-8：7 位地址格式

图 4-9 展示了主机发送数据到从机的过程。

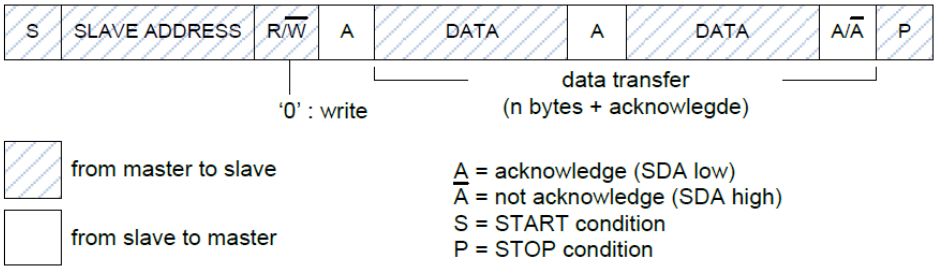

图 4-9： 主机发送数据

图 4-10 展示了主机从从机读取数据的过程。

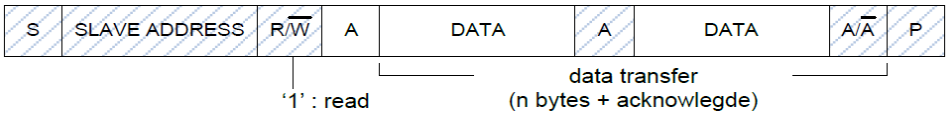

图 4-10：主机读取数据

图 4-11 展示了一个主机与从机的通讯过程。

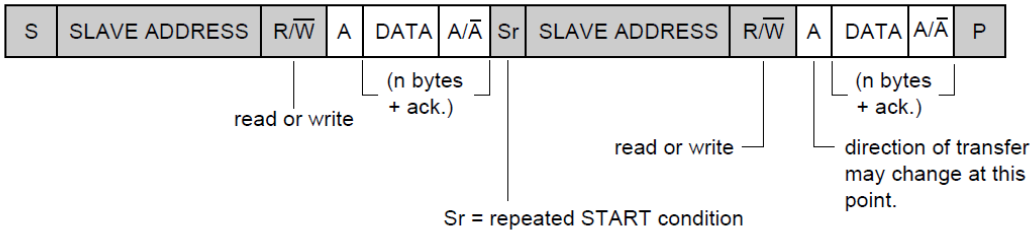

图 4-11：主机与从机的通讯过程

5. 工作模式

I2C 模块可根据 I2C 总线上的状态切换主机/从机模式。

5.1 主机模式

通过 I2C 主控模式定时周期寄存器(I2CMTP) 配置 I2C 模块的 SCL 时钟。

5.1.1 控制命令 1（空闲状态）

I2C 模块复位后，主机模式处于空闲状态，IDLE(I2CMSR[Bit5])为 1。

注：以下样例在 I2C 中断中清除 I2C 中断标志，因此 I2CMIF 位先置位（高电平），然后程序进入中断并清除 I2CMIF（低电平）。

1. START +（从机地址+写）+发送 1 字节数据

R/S	ACK	STOP	START	RUN	说明
0	x	0	1	1	(1) 地址+R/S = 从机地址 + 读/写，赋值给 I2CMSA 寄存器 (2) ACK 任意值 (3) 操作方法 : I2CMCR = 0x3

例：

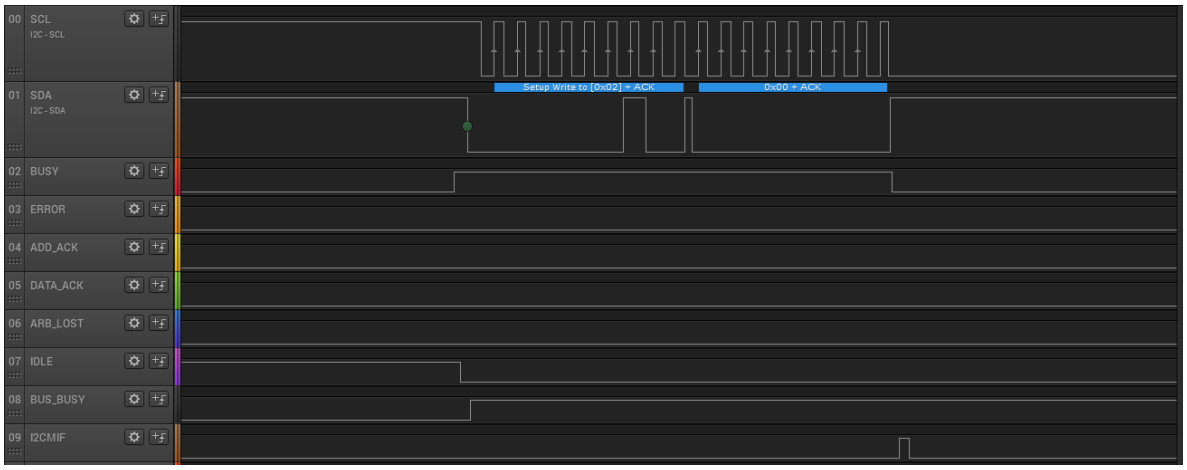

图 5-1：START +（从机地址+写）+发送 1 字节数据

开启发送后：

1) BUSY 位置位，发送完成后 BUSY 自动清零。
2) IDLE 位清零。
3) BUS_BUSY 位在监测到 START 信号后置位。
4) 发送完成后 I2CMIF 置位，产生中断。

2. START+（从机地址+写）+发送 1 字节数据+ STOP

R/S	ACK	STOP	START	RUN	说明
0	x	1	1	1	操作方法 : I2CMCR = 0x7

例：

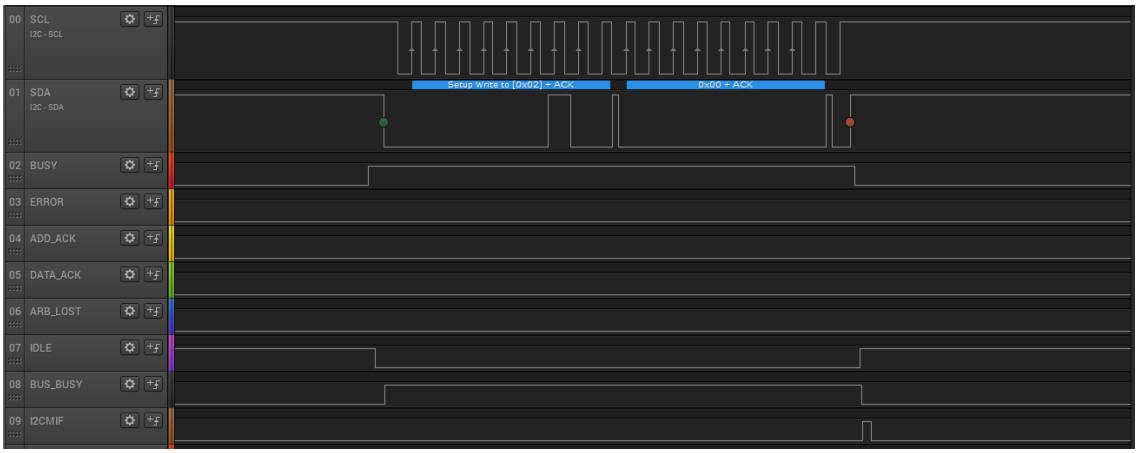

图 5-2：START+（从机地址+写）+发送 1 字节数据+ STOP

发送完 STOP 后，BUSY 位清零、IDLE 位清零、BUS_BUSY 位清零，I2CMIF 置位并产生中断。

3. START+（从机地址+读）+读取 1 字节数据+ NACK

R/S	ACK	STOP	START	RUN	说明
1	0	0	1	1	操作方法 : I2CMCR = 0x3

例：

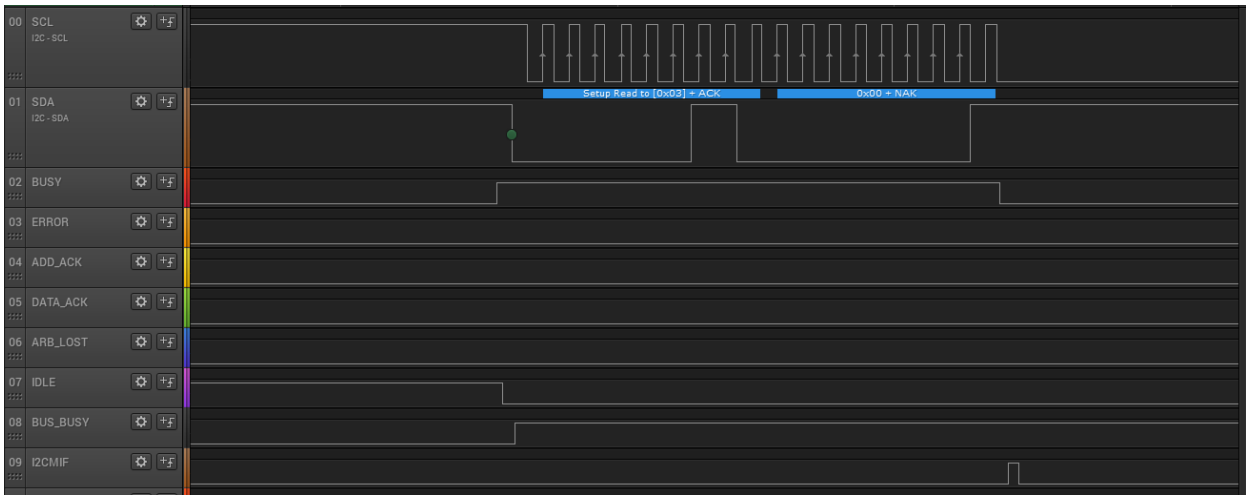

图 5-3：START +（从机地址+读）+读取 1 字节数据+ NACK

4. START+（从机地址+读）+读取 1 字节数据+ ACK

R/S	ACK	STOP	START	RUN	说明
1	1	0	1	1	操作方法 : I2CMCR = 0xb

例：

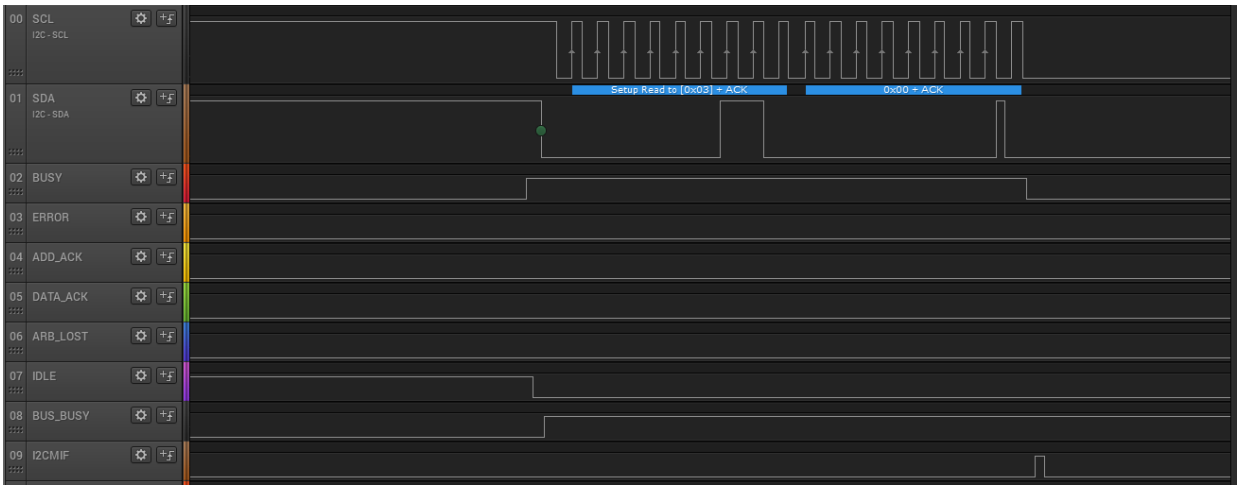

图 5-4：START +（从机地址+读）+读取 1 字节数据+ ACK

5. START+（从机地址+读）+读取 1 字节数据+NACK+STOP

R/S	ACK	STOP	START	RUN	说明
1	0	1	1	1	操作方法 : I2CMCR = 0x7

例：

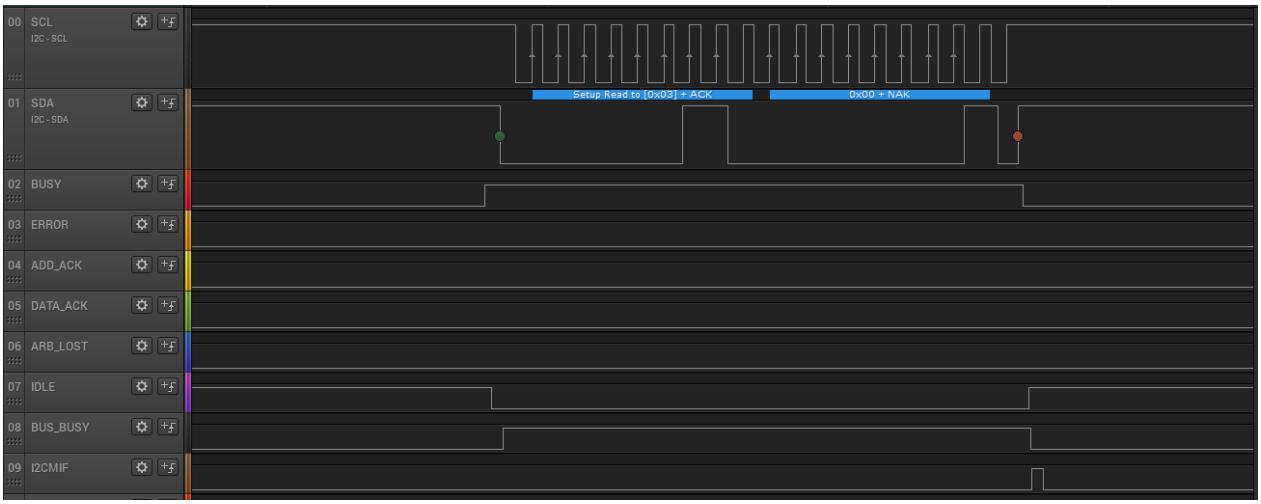

图 5-5：START +（从机地址+读）+读取 1 字节数据+NACK+STOP

6. 非法操作

1) START+（从机地址+读）+读取 1 字节数据+ACK+STOP

R/S	ACK	STOP	START	RUN	说明
1	1	1	1	1	操作方法 : I2CMCR = 0xF

例：

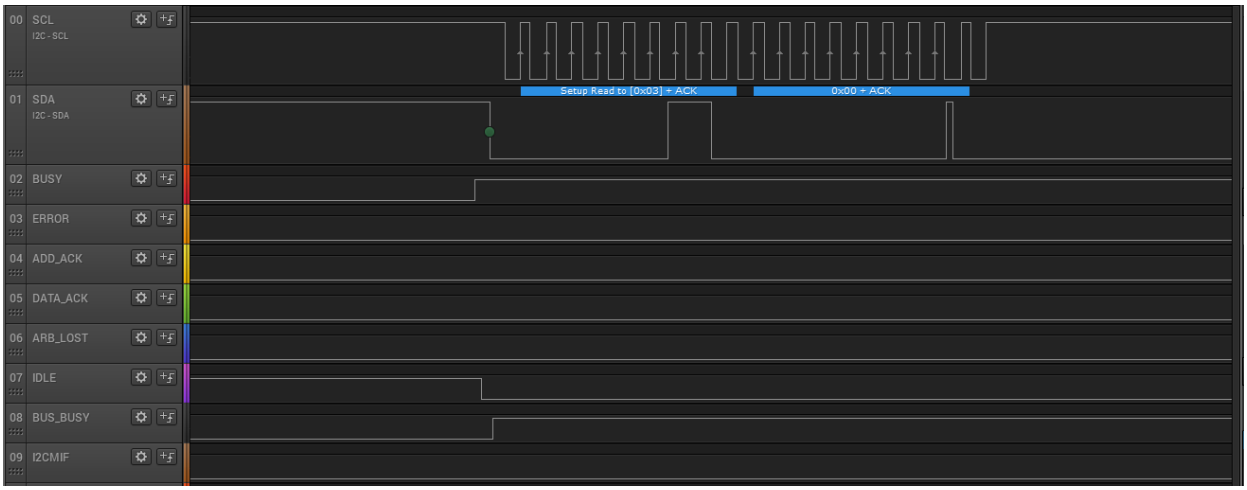

图 5-6：START +（从机地址+读）+读取 1 字节数据+ACK+STOP

I2C 发送数据完成后会立即发送 STOP，但从机占用了 SDA，故不能成功发送 STOP 信号。因此主机不能产生中断。

5.1.2 控制命令 2（发送状态）

发送状态也可理解为 主机往从机写数据。

1. 发送 1 字节数据

R/S	ACK	STOP	START	RUN	说明
0	0	0	0	1	操作方法 : I2CMCR = 0x1

例：

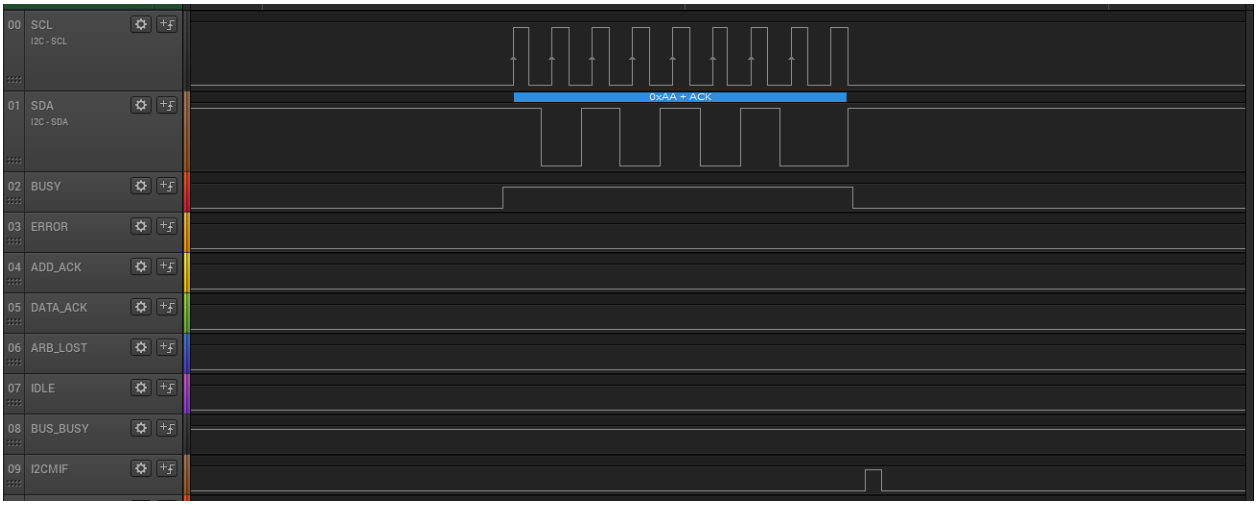

图 5-7：发送 1 字节数据

2. 发送 STOP 信号

R/S	ACK	STOP	START	RUN	说明
0	0	1	0	0	操作方法 : I2CMCR = 0x4

例：

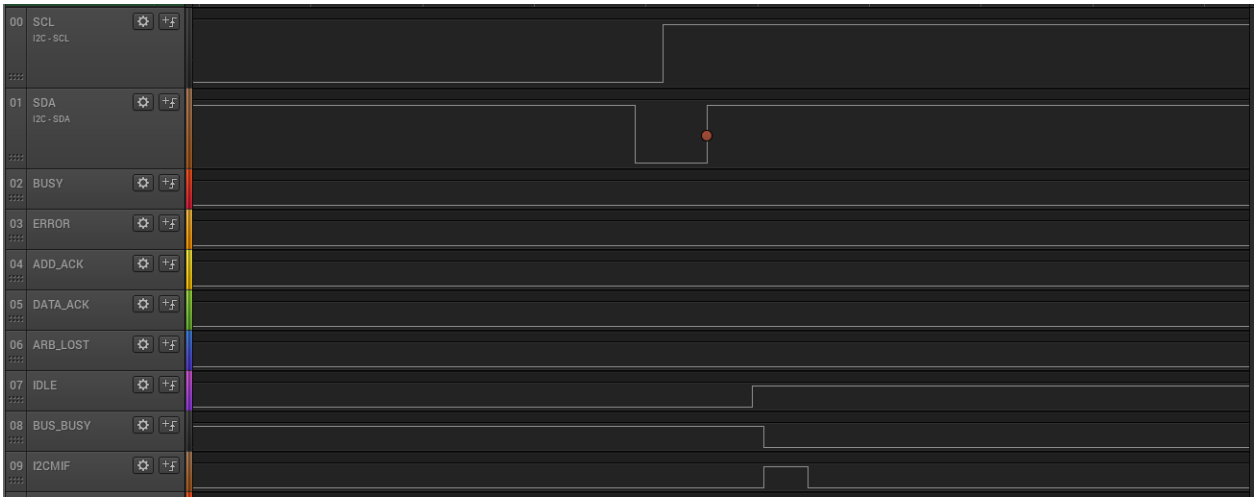

图 5-8：发送 STOP 信号

3. 发送 1 字节数据+STOP 信号

R/S	ACK	STOP	START	RUN	说明
0	0	1	0	1	操作方法 : I2CMCR = 0x5

例：

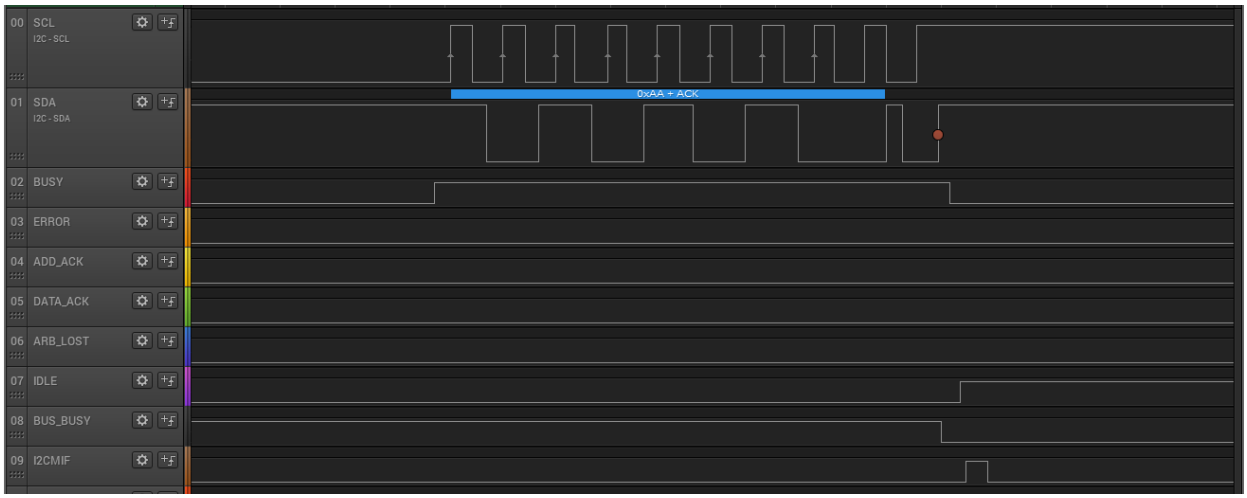

图 5-9：发送 1 字节数据+STOP 信号

4. 发送 RSTART + （从机地址+写）+ 发送 1 字节数据

R/S	ACK	STOP	START	RUN	说明
0	0	0	1	1	操作方法 : I2CMCR = 0x3

例：

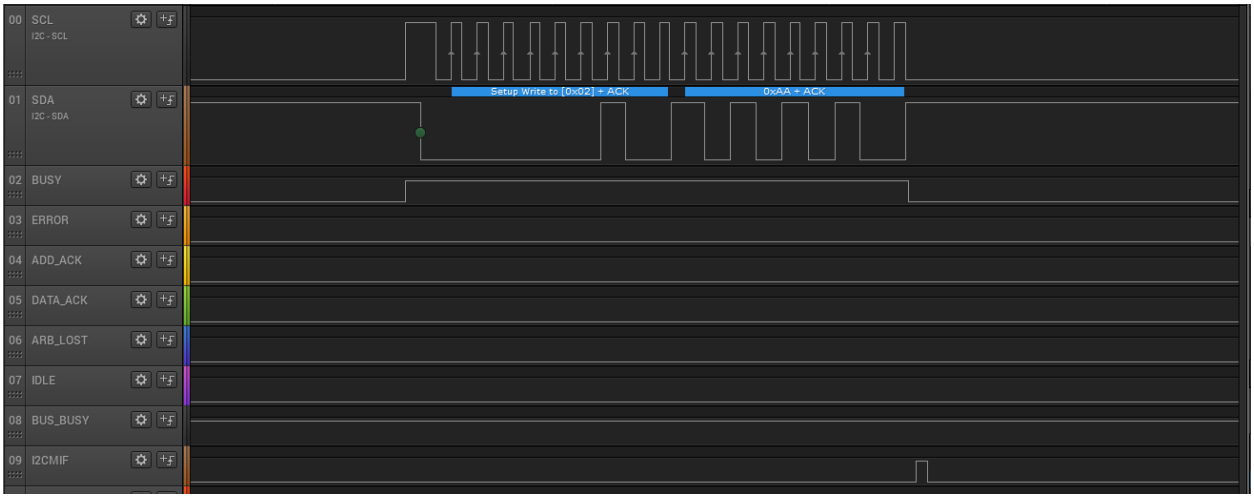

图 5-10：发送 RSTART + （从机地址+写）+ 发送 1 字节数据

5. 发送 RSTART + （从机地址+写）+ 发送 1 字节数据+STOP 信号

R/S	ACK	STOP	START	RUN	说明
0	0	1	1	1	操作方法：I2CMCR = 0x7

例：

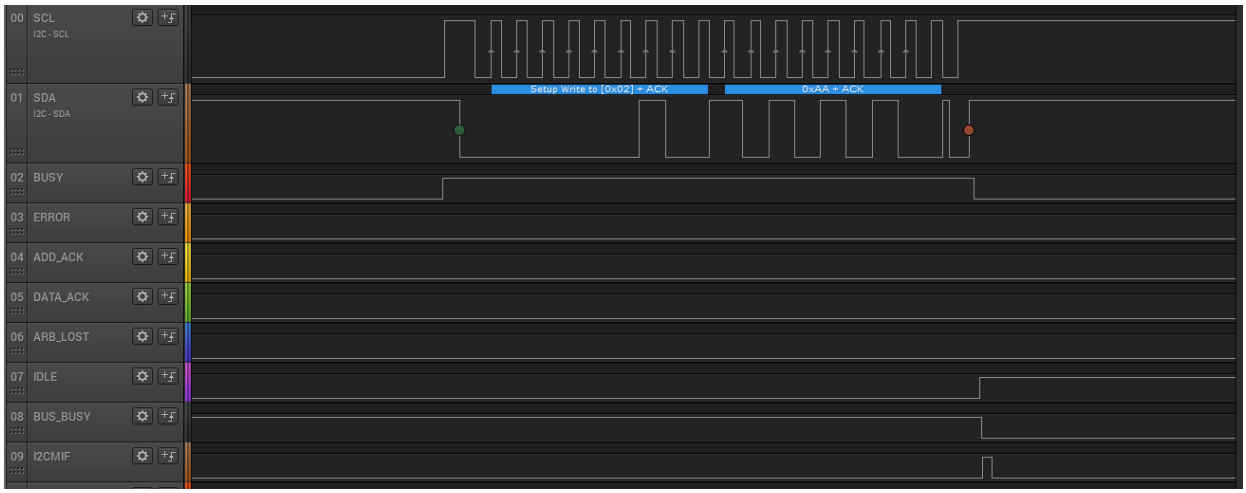

图 5-11：发送 RSTART + （从机地址+写）+ 发送 1 字节数据+STOP 信号

6. 发送 RSTART+从机地址 + 读取 1 字节数据+NACK

R/S	ACK	STOP	START	RUN	说明
1	0	0	1	1	操作方法：I2CMCR = 0x3

例：

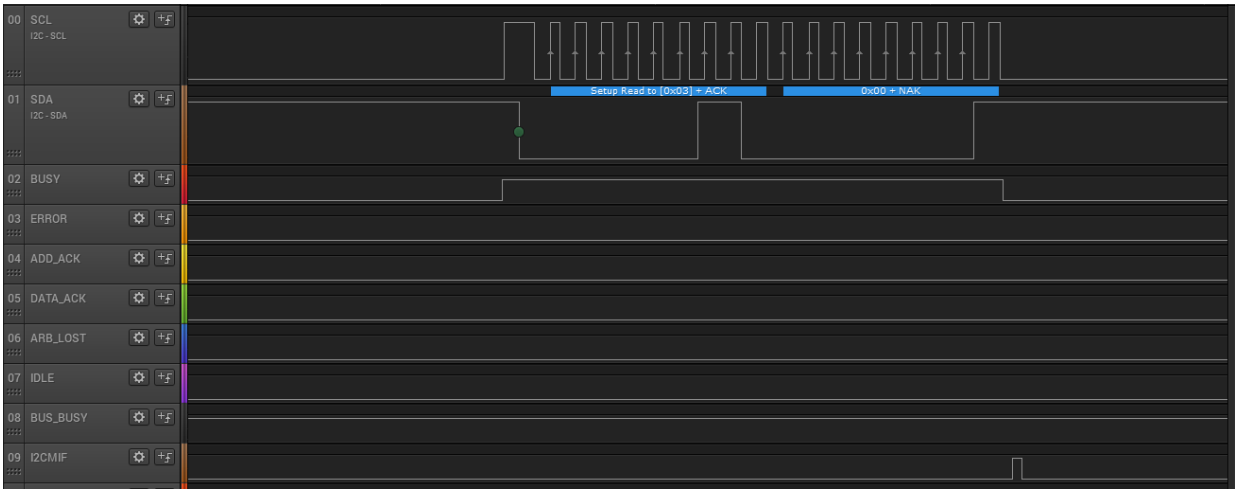

图 5-12：发送 RSTART+从机地址 + 读取 1 字节数据+NACK

7. 发送 RSTART+从机地址 + 读取 1 字节数据+NACK+ STOP

R/S	ACK	STOP	START	RUN	说明
1	0	1	1	1	操作方法 : I2CMCR = 0x7

例：

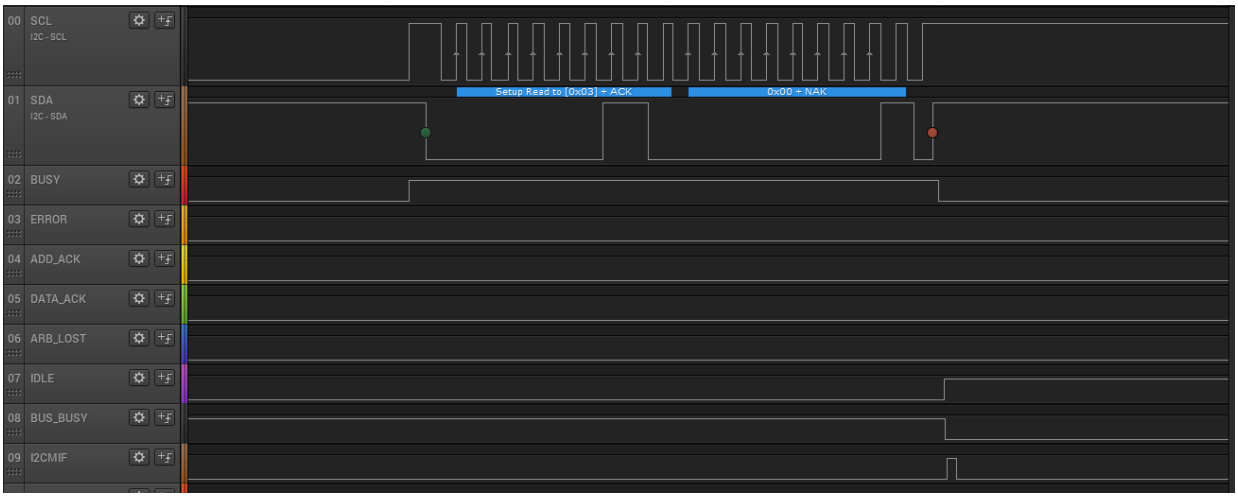

图 5-13：发送 RSTART+从机地址 + 读取 1 字节数据+NACK+ STOP

8. 发送 RSTART+从机地址 + 读取 1 字节数据+ACK

R/S	ACK	STOP	START	RUN	说明
1	1	0	1	1	操作方法：I2CMCR = 0xb

例：

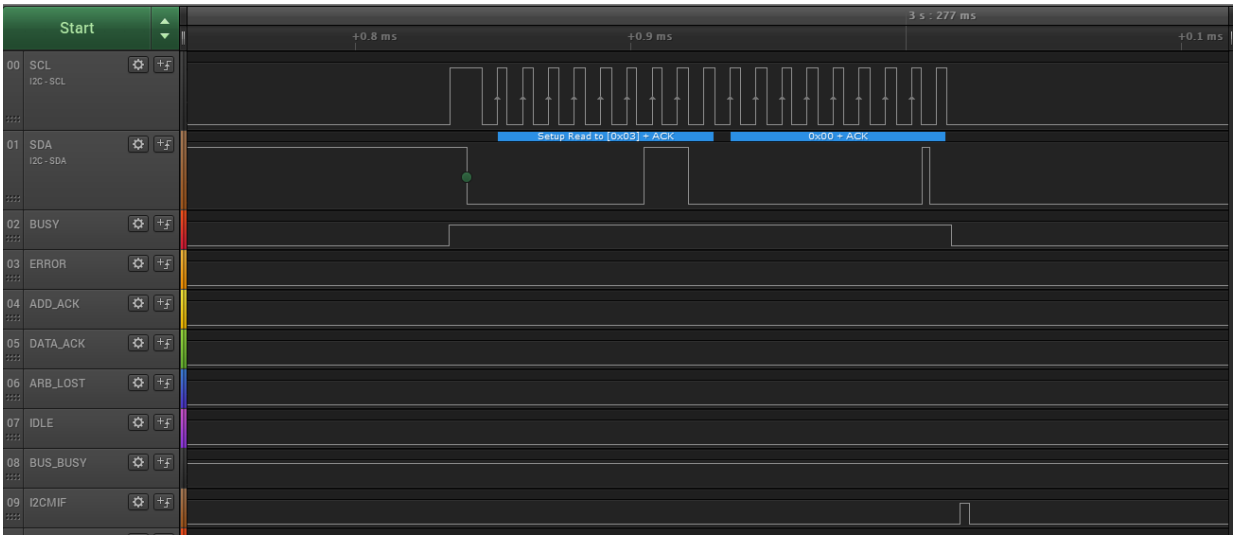

图 5-14：发送 RSTART+从机地址 + 读取 1 字节数据+ACK

9. 非法操作

发送 RSTART+从机地址 + 读取 1 字节数据+ACK+STOP

R/S	ACK	STOP	START	RUN	说明
1	1	1	1	1	操作方法：I2CMCR = 0xF

例：

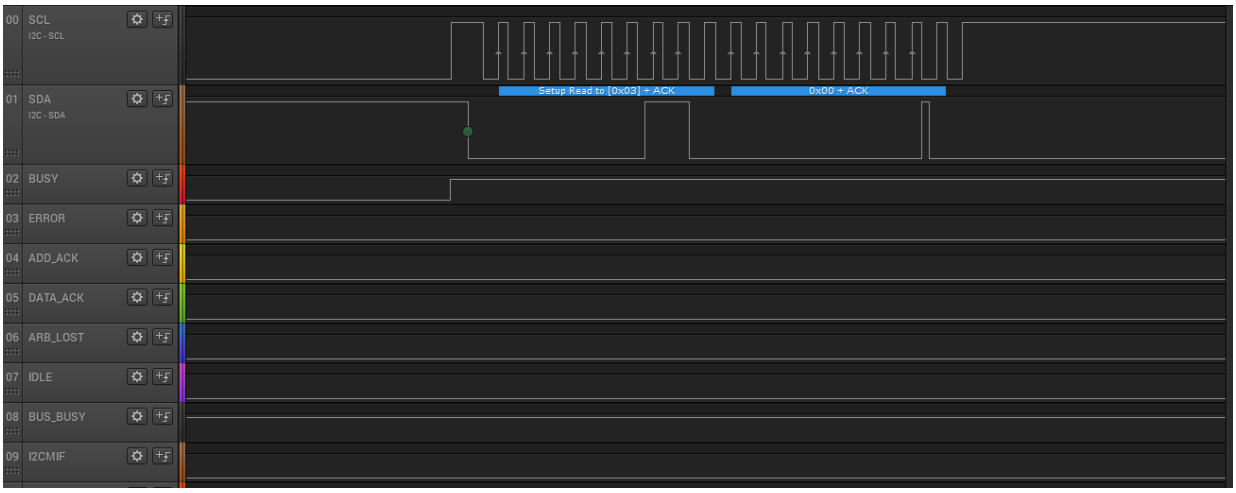

图 5-15：发送 RSTART+从机地址 + 读取 1 字节数据+ACK+STOP

5.1.3 主控命令 3（接收状态）

主控接收状态也可理解为主机在读取从机数据。

1. 读取 1 字节数据+NACK

R/S	ACK	STOP	START	RUN	说明
1	0	0	0	1	操作方法：I2CMCR = 0x1

例：

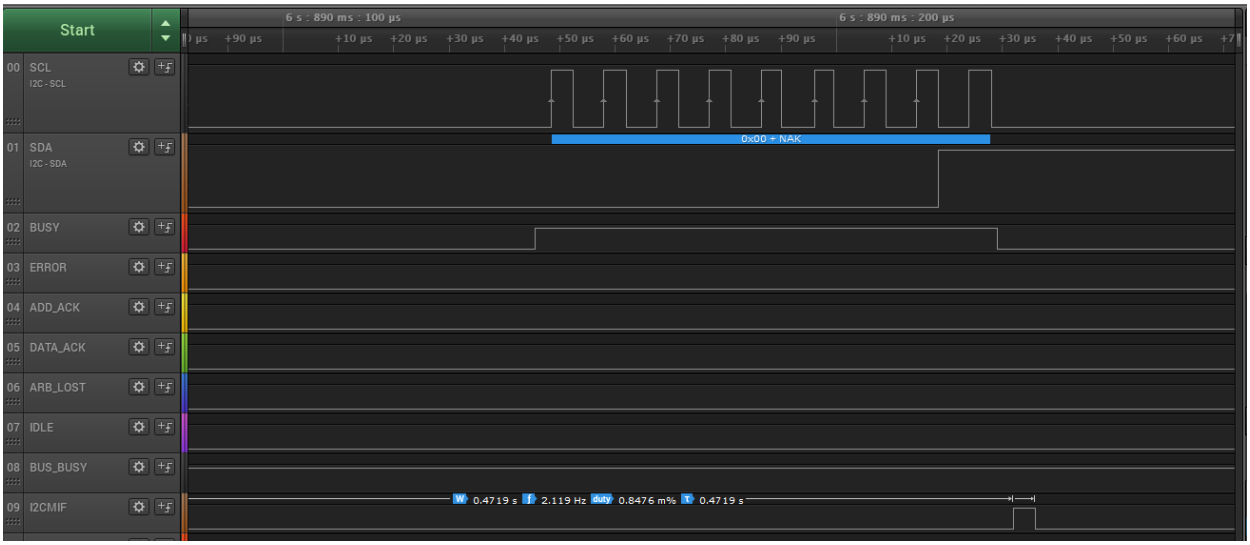

图 5-16：读取 1 字节数据+NACK

2. 发送 STOP

R/S	ACK	STOP	START	RUN	说明
1	0	1	0	0	操作方法：I2CMCR = 0x4

例：

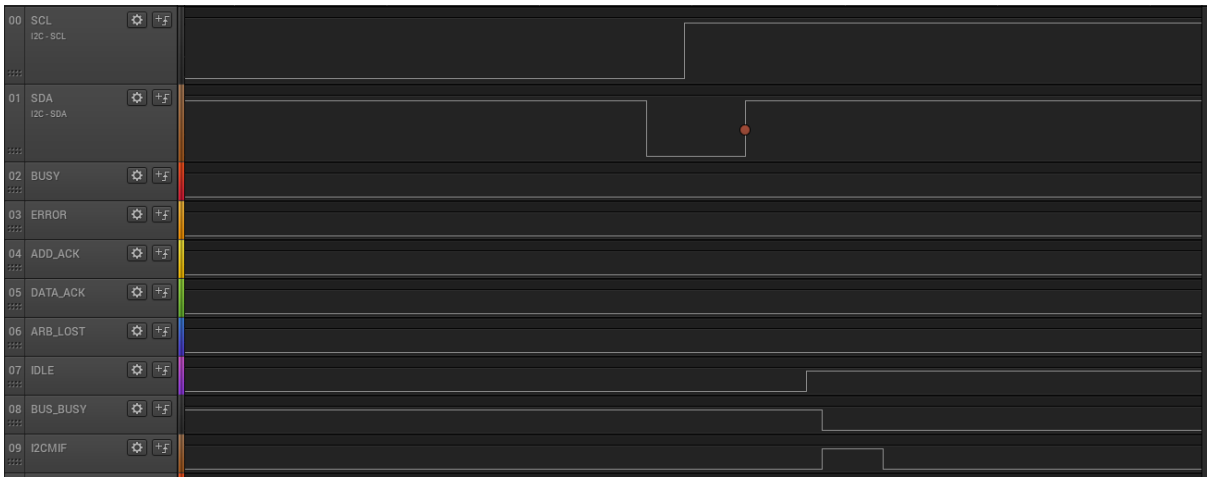

图 5-17：发送 STOP

3. 读取 1 字节数据+STOP

R/S	ACK	STOP	START	RUN	说明
1	0	1	0	1	操作方法：I2CMCR = 0x5

例：

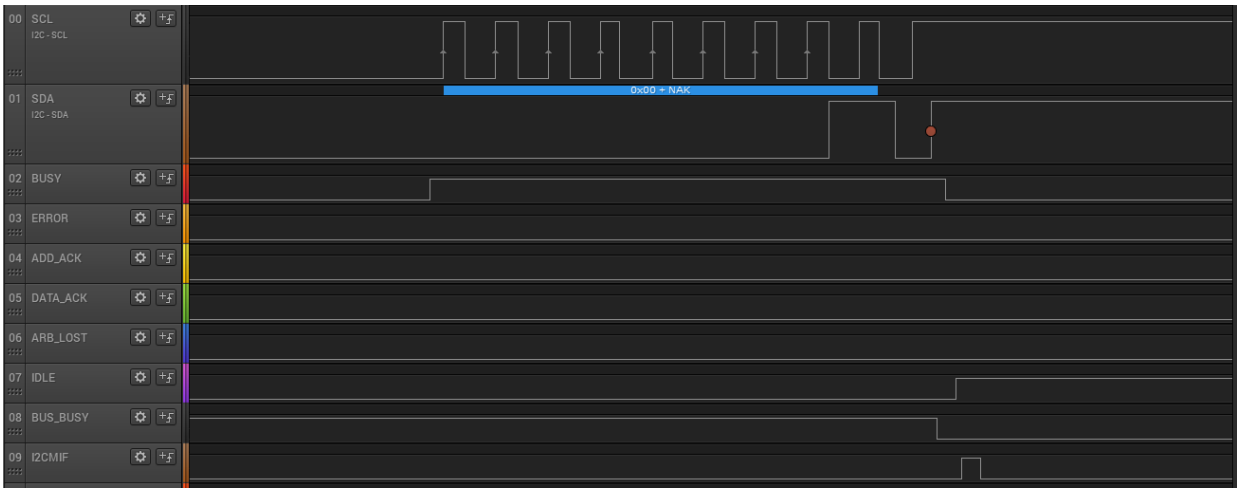

图 5-18：读取 1 字节数据+STOP

4. 读取 1 字节数据+ ACK

R/S	ACK	STOP	START	RUN	说明
1	1	0	0	1	操作方法：I2CMCR = 0x9

例：

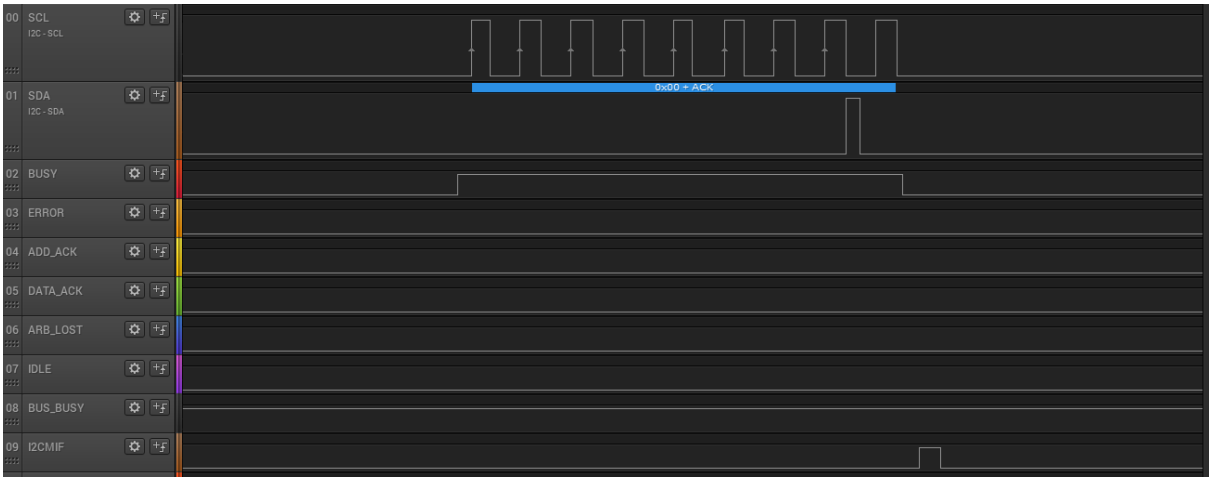

图 5-19：读取 1 字节数据+ ACK

5. 发送 RSTART+（从机地址+读）+读取 1 字节数据+ NACK

R/S	ACK	STOP	START	RUN	说明
1	0	0	1	1	操作方法：I2CMCR = 0x3

例：

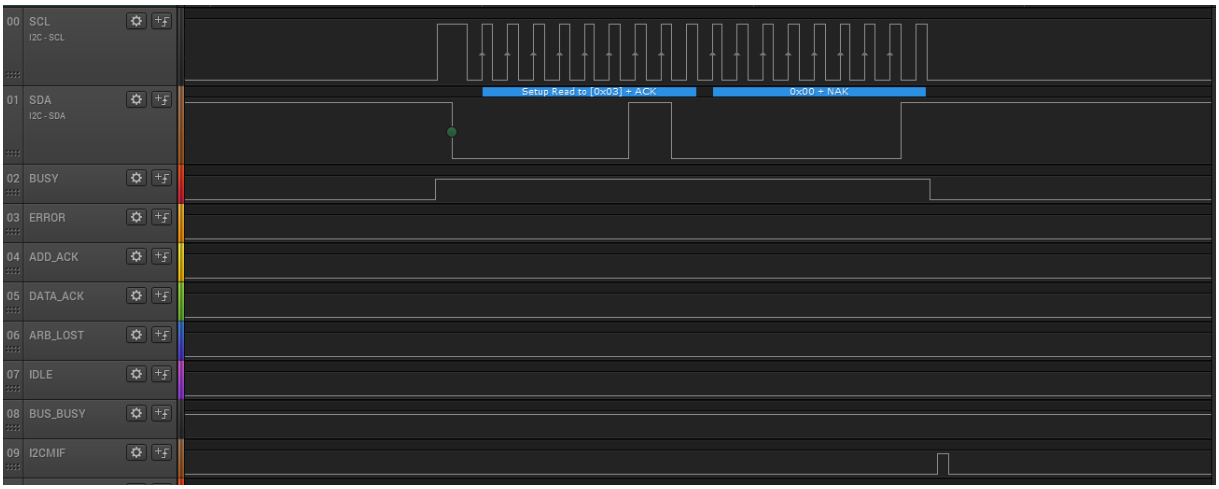

图 5-20：发送 RSTART+（从机地址+读）+读取 1 字节数据+ NACK

6. 发送 RSTART+（从机地址+读）+读取 1 字节数据+ NACK + STOP

R/S	ACK	STOP	START	RUN	说明
1	0	1	1	1	操作方法：I2CMCR = 0x7

例：

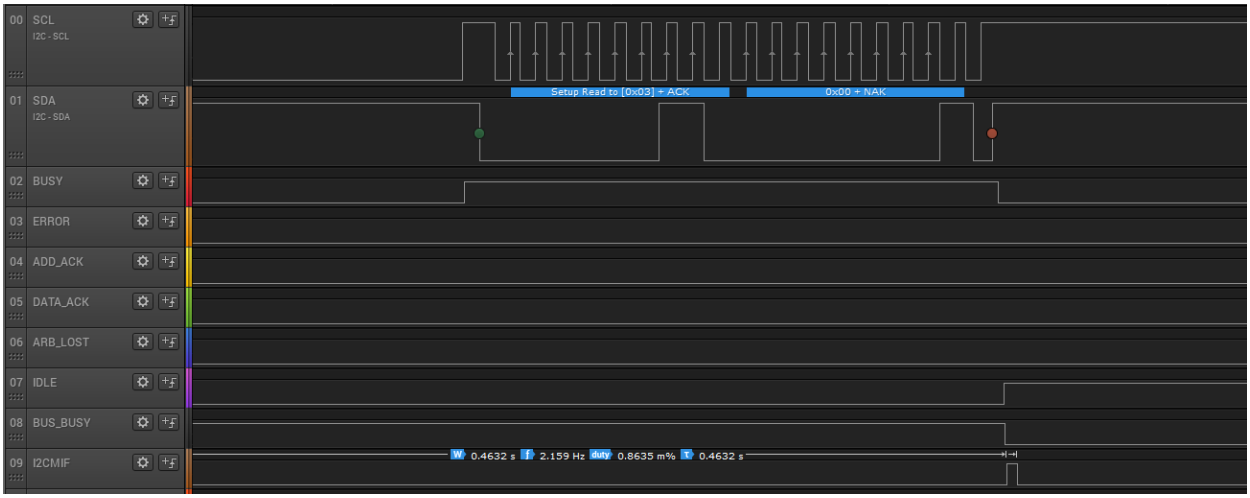

图 5-21：发送 RSTART+（从机地址+读）+读取 1 字节数据+ NACK + STOP

7. 发送 RSTART+（从机地址+读）+读取 1 字节数据+ ACK

R/S	ACK	STOP	START	RUN	说明
1	1	0	1	1	操作方法：I2CMCR = 0xb

例：

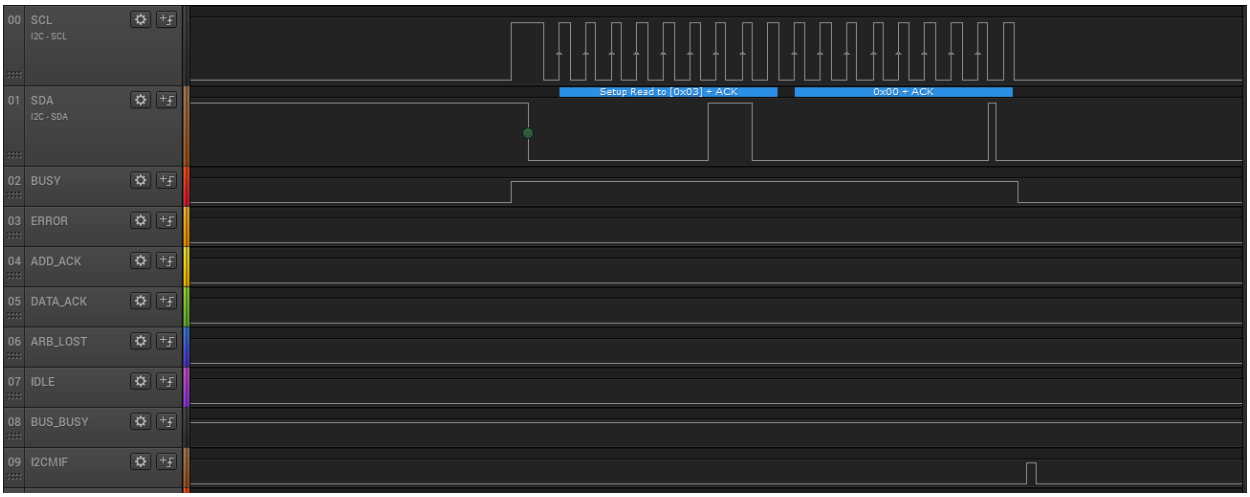

图 5-22：发送 RSTART+（从机地址+读）+读取 1 字节数据+ ACK

8. 发送 RSTART+（从机地址+写）+发送 1 字节数据

R/S	ACK	STOP	START	RUN	说明
0	0	0	1	1	操作方法：I2CMCR = 0x3

例：

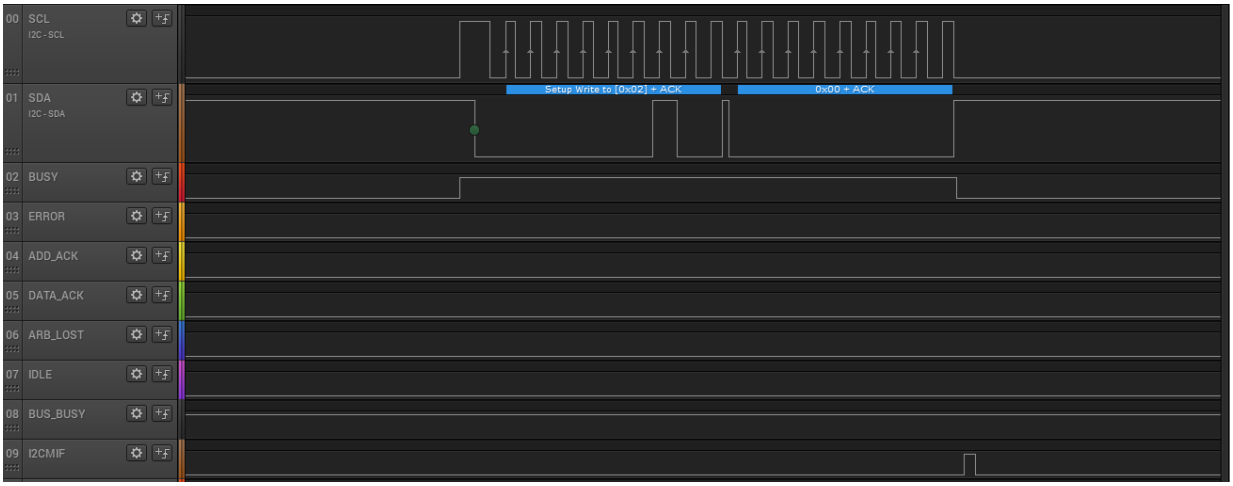

图 5-23：发送 RSTART+（从机地址+写）+发送 1 字节数据

9. 发送 RSTART+（从机地址+写）+发送 1 字节数据+STOP

R/S	ACK	STOP	START	RUN	说明
0	0	1	1	1	操作方法：I2CMCR = 0x7

例：

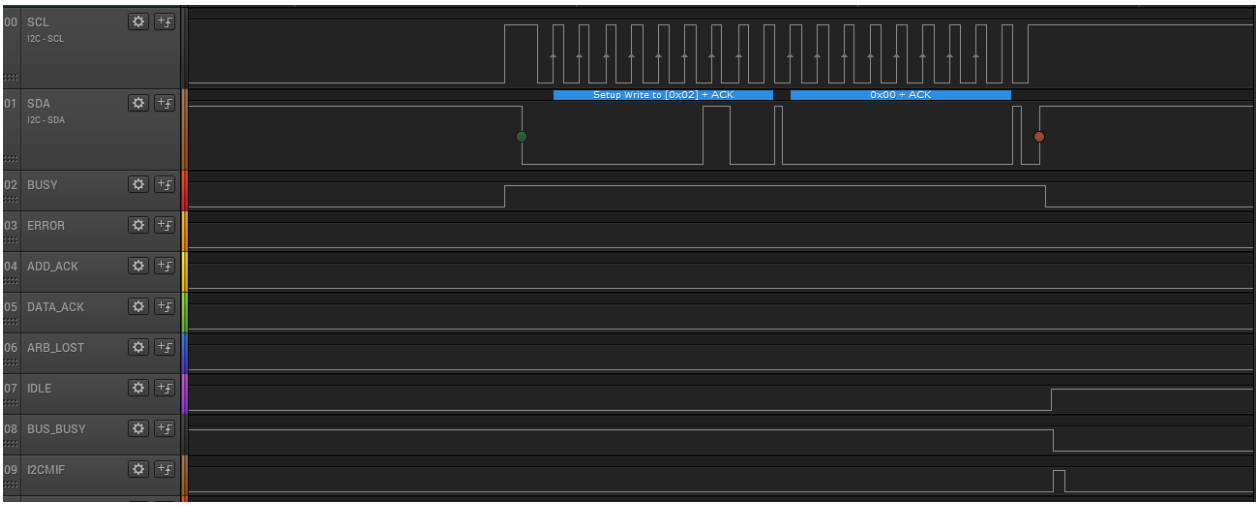

图 5-24：发送 RSTART+（从机地址+写）+发送 1 字节数据+STOP

5.1.4 异常状态

1. 从机地址未响应、从机回 NACK。

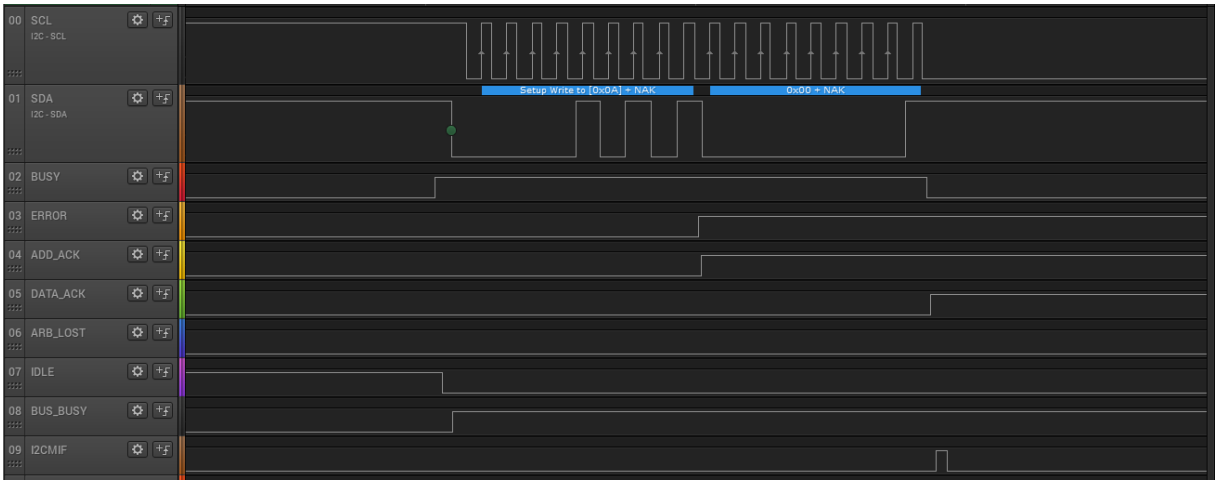

图 5-25：从机地址未响应、从机回 NACK

1) 寻址失败后 ADD_ACK 将会置位。
2) 发送数据，从机回 NACK，DATA_ACK 位会置位。
3) 清中断标志并不会清除状态位。
4) 重新寻址成功或者复位 I2C 主控模块 RSTS 置位 (I2CMCR[bit7])，错误状态位将会清除。

2. I2C 主控模式丢失主控权、仲裁冲突。

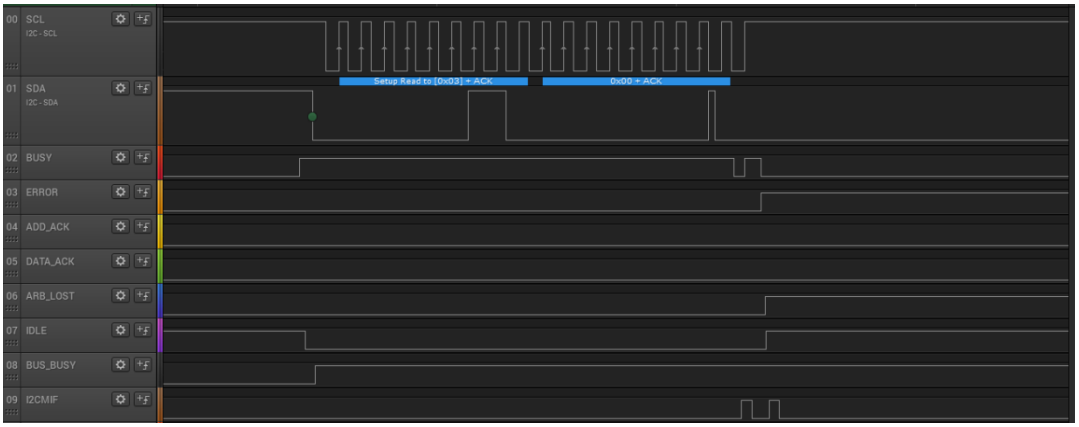

图 5-26：I2C 主控模式丢失主控权、仲裁冲突

当主机无法控制 SDA 时（上述描述的现象：主机读取从机时，主机发送了 ACK 信号，从机占据 SDA 控制权），主机又开始了重新发送起始位等操作，此时将会出现主控模式错误状态（丢失主控权），并且产生中断。

通过复位主控模式或者恢复正常通讯，丢失主控权的标志自动清除。

1) 复位主控模式

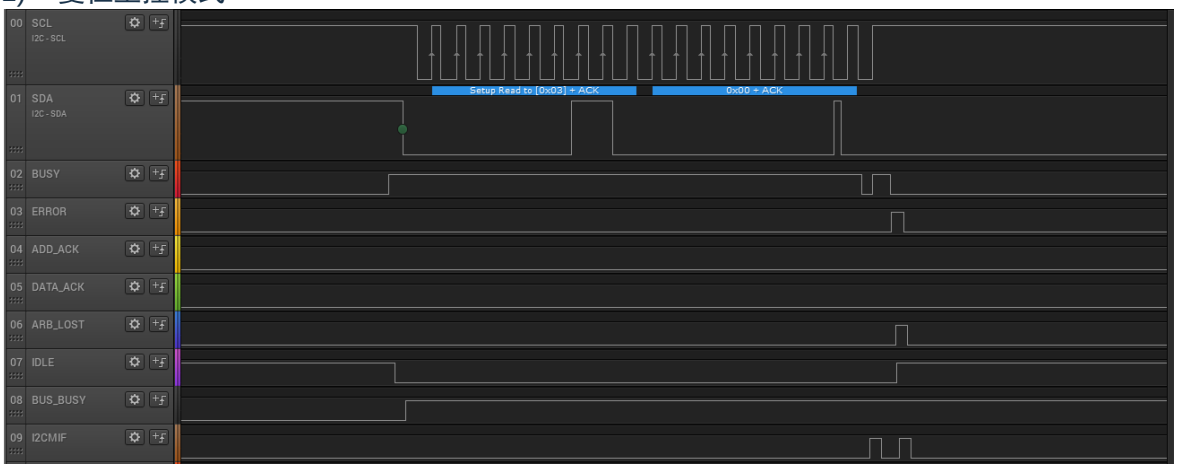

图 5-27：复位主控模式

2) 从机释放 SDA 控制权，主机恢复正常通讯

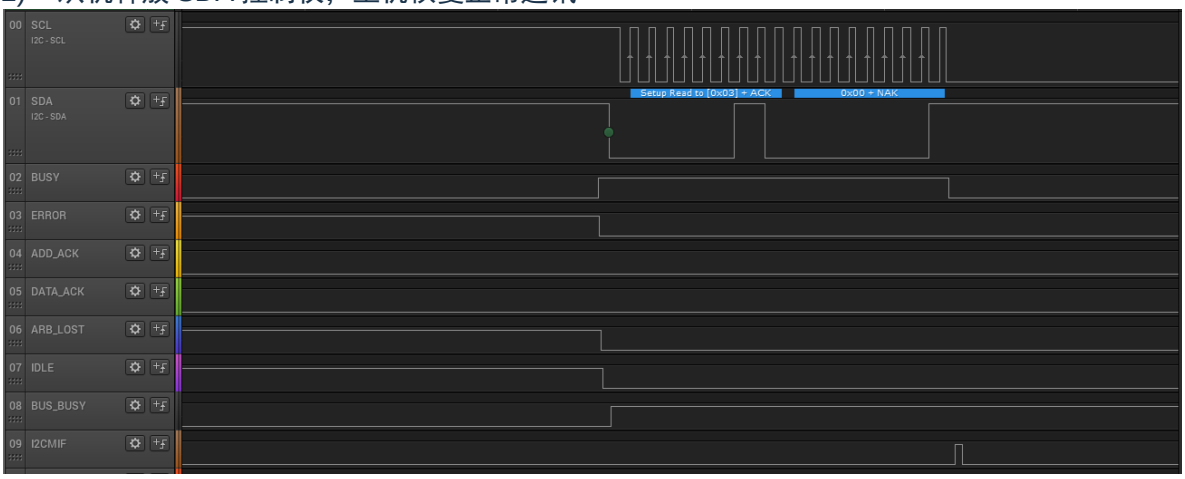

图 5-28：从机释放 SDA 控制权，主机恢复正常通讯

5.2 从机模式

从机模式下需要设置自身的从机地址（7 位）寄存器 I2CSADR。DA(I2CSCR[bit0])位设置为 1。

5.2.1 接收数据

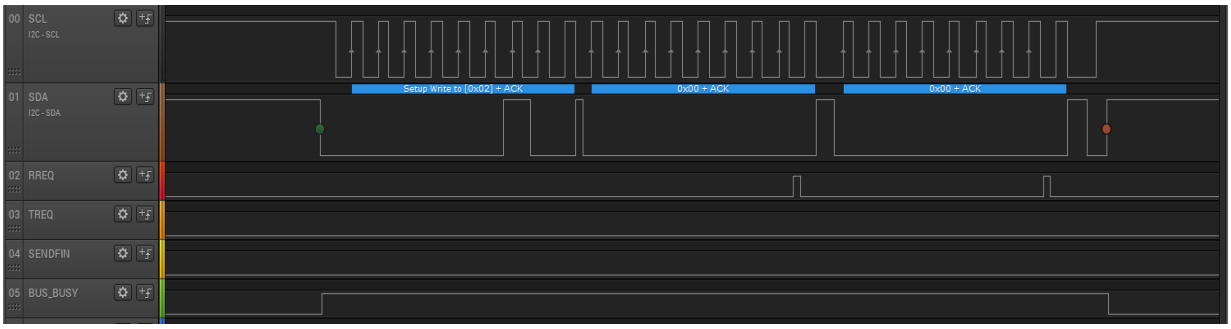

图 5-29：接收数据

1. 当从机被成功寻址并在接收到第一个数据后，RREQ 标志置位，读取 I2CSBUF 后 RREQ 标志自动清除。
2. 之后每成功接收到一个数据，RREQ 标志将会置位，读取 I2CSBUF 后 RREQ 标志自动清除。
3. BUS_BUSY 位在总线上出现启动信号时置位，当出现停止信号时自动清零。

5.2.2 发送数据

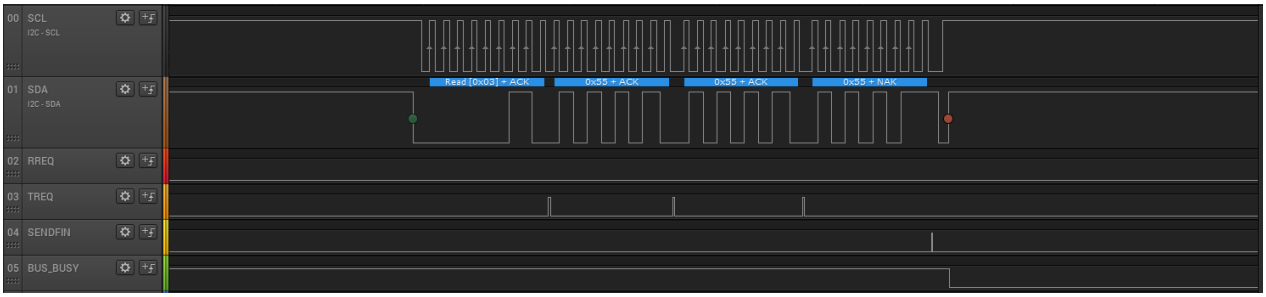

图 5-30：发送数据

1. 当从机被成功寻址后，TREQ 位被置位，写 I2CSBUF 寄存器后 TREQ 位自动清除。
2. 当从机接收到主机读取 + ACK 信号后，TREQ 位置位，写 I2CSBUF 寄存器后 TREQ 位自动清除。
3. 当从机接收到主机读取 + NACK 信号后，SENDFIN 位置位，读取 I2CSSR 寄存器后 SENDFIN 位自动清除。

6. 应用注意事项

6.1 IO 口配置

在使用 I2C 模块时需要对 IO 口进行功能复用，需要配置 PxxCFG 寄存器。不同的芯片的配置方式略有不同，以下列举了 8051 内核芯片的 I2C 模块 SDA、SCL 功能配置方法。

6.1.1 全功能复用

IO 口全功能复用的芯片

芯片系列	芯片型号	分类	IO 功能配置
CMS8S 系列	CMS8S3660	-	以 CMS8S6990 芯片为例： P00 配置为 SDA、P01 配置为 SCL： P00CFG = 0x0D。 P01CFG = 0x0C（具体数值请参考对应的芯片参考手册）。
	CMS8S3680	-	
	CMS8S69xx	CMS8S6980	
		CMS8S6990	
	CMS8S5880	-	
	CMS8S5885	-	
	CMS8S589x	CMS8S5895	
		CMS8S5897	
		CMS8S5898	
		CMS8S5899	
混合信号-无刷电机系列	CMS8M1010	-	
	CMS8M35xx	CMS8M3510	
		CMS8M3524	
		CMS8M3533	
混合信号-高精度测量系列	CMS8H5109	-	
	CMS8H5120	-	

6.1.2 非全功能复用

IO 口非全功能复用的芯片

芯片系列	芯片型号	分类	IO 功能配置
CMS8S 系列	CMS8S588x	CMS8S5887	以 CMS8S588x 芯片为例： P01 配置为 SDA、P00 配置为 SCL： P00CFG = 0x02。 P01CFG = 0x02。 PS_SCL = 0x00。 PS_SDA = 0x01。 （具体数值请参考对应的芯片参考手册）。 注：使用 I2C 功能（主控模式、从机模式），必须完全配置 PS_SCL 和 PS_SDA 寄存器。
		CMS8S5888	
		CMS8S5889	
	CMS8S78xx	CMS8S7885	
		CMS8S7895	
CMS80F 系列	CMS80F261x	CMS80F2618	
		CMS80F2619	
		CMS80F261A	
		CMS80F261B	
	CMS80F253x	CMS80F2538	
		CMS80F2539	
	CMS80F751x	CMS80F7518	
		CMS80F7519	
混合信号-高精度测量系列	CMS8H5101L	-	
	CMS8H5106	-	

6.2 Clock Stretching

6.2.1 从机

I2C 从机模式具有 Clock Stretching 功能，当从机需要更多时间保存或者准备下一个要发送的数据时，从机将会拉低 SCL，迫使主机进入等待状态，直到从机准备好。

1. 从机及时响应

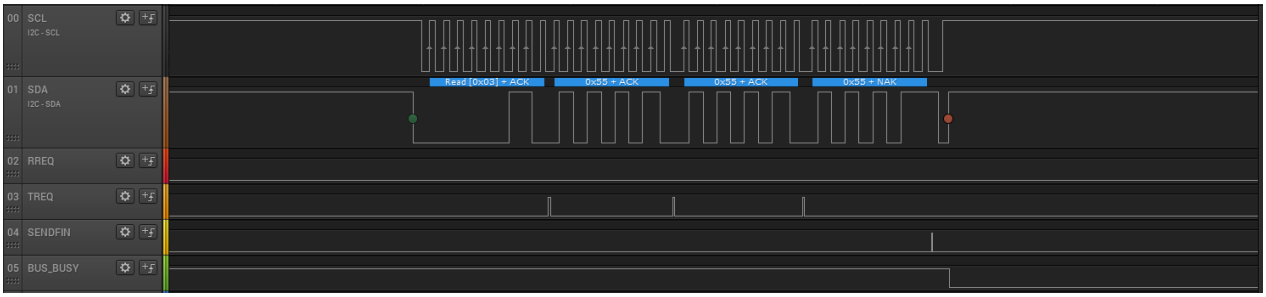

图 6-1：从机及时响应

2. Clock Stretching

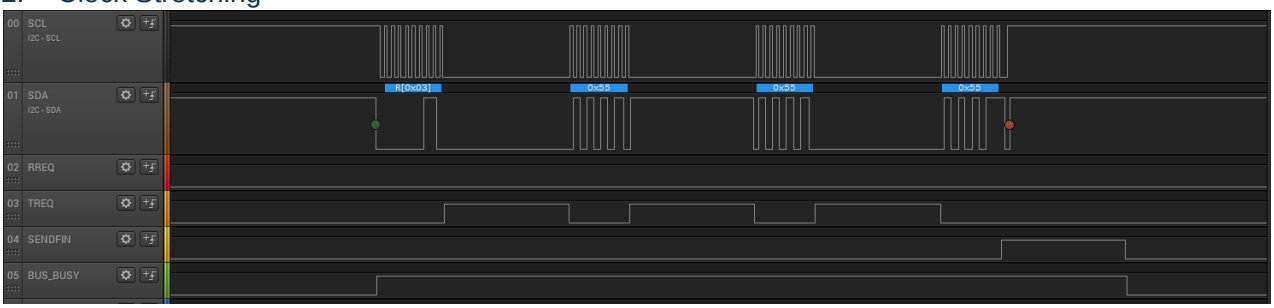

图 6-2：Clock Stretching

1) 从机被寻址或者 接收到主机 读取+ACK 信号后，TREQ 置位 。写 I2CSBUF 寄存器后 TREQ 位自动清零，从机将会释放 SCL。
2) 当从机 接收到主机 读取+NACK 信号后，SENDFIN 位将置位，因为从机将不会再发送数据，则从机不会拉低 SCL。

6.2.2 主机

当从机拉低 SCL 时，主机将会进入等待状态。若主机不具备此功能时，将会出现丢失 SCL 的现象。

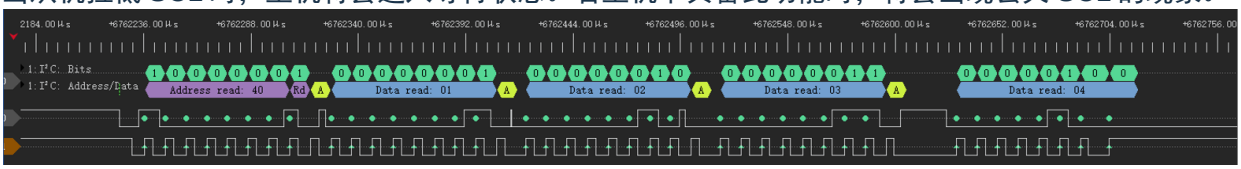

图 6-3：丢失 SCL

如上图所示，在最后 1 字节发送中，总线上丢失了 2 个 SCL 时钟。

6.3 SENDFIN 标志位

SENDFIN 标志位表示 I2C 从机的模式下检测到了主机发送的 NACK 信号。当读取 I2CSSR 寄存器后，硬件会自动清

零 SENDFIN 标志。在使用中需要注意避免一直读取 I2CSSR 寄存器获取 SENDFIN 标志，例如等待 SENDIFN 标志置位：while((I2CSSR&0x4)==0)，此操作方式会存在 I2C 接收到 NACK 信号时硬件置位 SENDFIN 标志 与 软件读取 I2CSSR 寄存器硬件清除 SENDIFN 标志冲突的情况，并且当两者同时发生时 SENDFIN 标志将会保持为 0。

读取 SENDIFN 推荐的操作方式：

1. 开启 I2C 中断（I2CIE（I2C 中断使能位）= 1）
   1) 在进入 I2C 中断服务程序后，使用临时变量存储 I2CSSR 的值，然后再对临时变量的值进行判断。
2. 不开启 I2C 中断（I2CIE（I2C 中断使能位）= 0）
   1) 判断 I2C 中断标志是否置位。
   2) 若 I2C 中断标志为 1，使用临时变量存储 I2CSSR 的值，然后再对临时变量的值进行判断。

6.4 ACK 信号引起的异常

当从机接收到主机发送的“读取+ACK”信号后，从机将会占用 SDA 的控制权。当从机接收到主机发送的“读取+NACK”信号后，从机将会释放 SDA 的控制权。若环境中存在干扰，主机发送的 NACK 信号被从机误识别为 ACK 信号，则主机将会失去对 SDA 的控制权，因此主机将无法再正常发送 STOP、START 信号，从而出现通讯“卡死”的现象。

例如：

1. 正常通讯

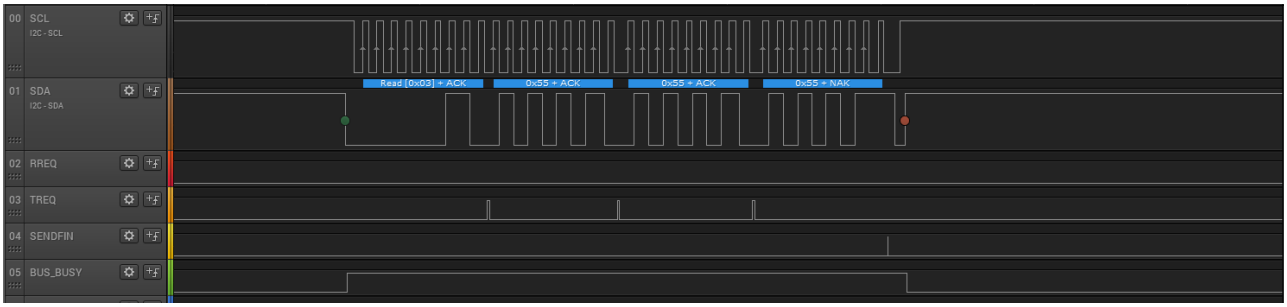

图 6-4：正常通讯

2. 模拟环境干扰导致从机误把 NACK 识别为 ACK 信号

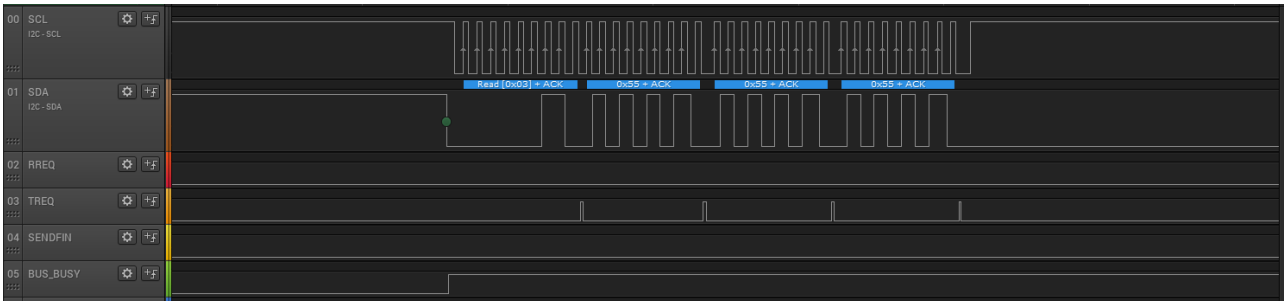

图 6-5：模拟异常现象

当从机控制 SDA 时，主机无法完成 STOP 信号发送。

注：也存在例外，当从机发送的数据最高位为 1 时，主机将会完成 STOP 信号发送，当从机接收到 STOP 信号后，将会停止此次发送，等待主机下次寻址。

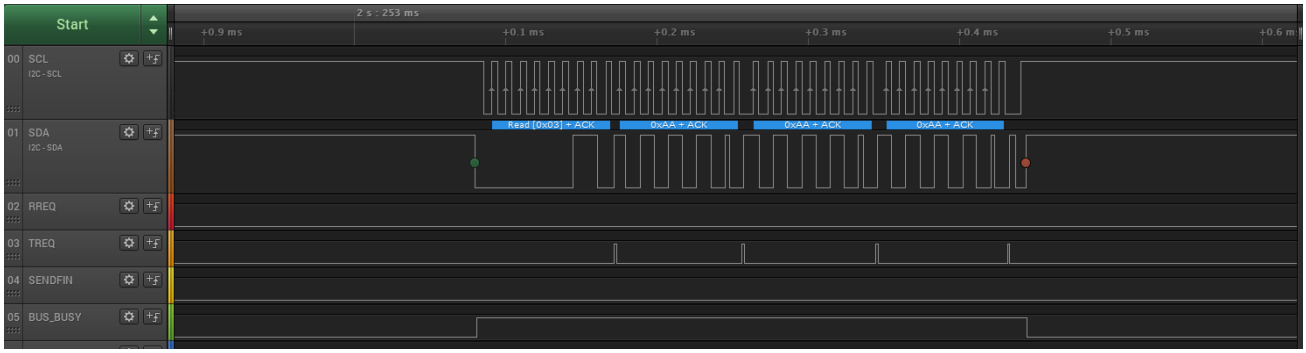

图 6-6：异常情况

解决办法：

1. 修改主机控制方法（以 CMS8S6990 为例）。

   暂停 I2C 主机模式控制，转而直接使用 GPIO，模拟一次 读取+NACK 信号时序发送。以下以 CMS8S6990 为例。

   1) 配置 SDA、SCL 的 GPIO 为开漏输出模式，SDA 输出 1，SCL 输出 1。
   2) 修改 SDA、SCL 的 IO 模式为 GPIO 模式，PxxCFG = 0x00；
   3) 在发送时保持 SDA 输出 1，SCL 输出 8 个 SCL 波形。
   4) 修改 SDA、SCL 的 IO 模式为 I2C 模式。

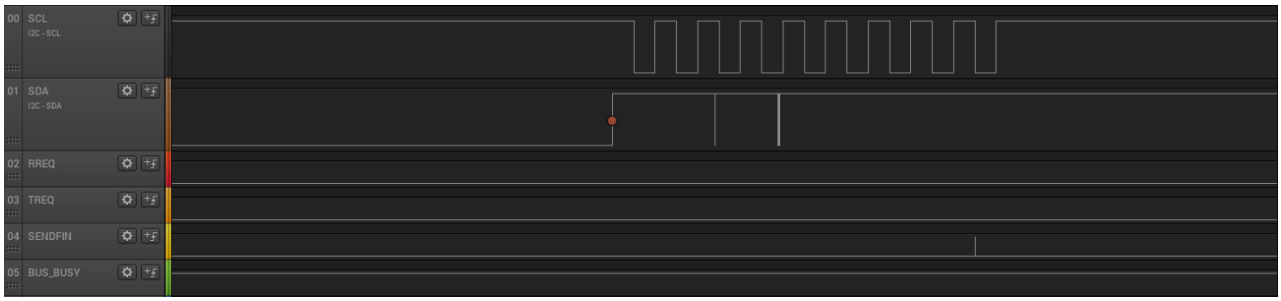

图 6-7：解决办法

从上图可看出，当完成 SCL 发送后，SENDFIN 位置 1，读取 I2CSSR 寄存器后，该位自动清除。当从机释放 SDA 后，主机即可再次控制 SDA。

注：图 6-7 解析器将出现的红点解析为 STOP 信号，但 BUS_BUSY 位仍置 1，我们一般以 BUS_BUSY 位的状态为参考。

2. 修改从机控制方法

若从机长时间没有检测到主机发送的信息时（检测 SENDFIN 位），可配置寄存器复位从机模式，从而释放对 SDA 的控制权。

6.5 时钟频率

6.5.1 I2C 模块 SCL 时钟频率

I2C 时序分解如下图所示：

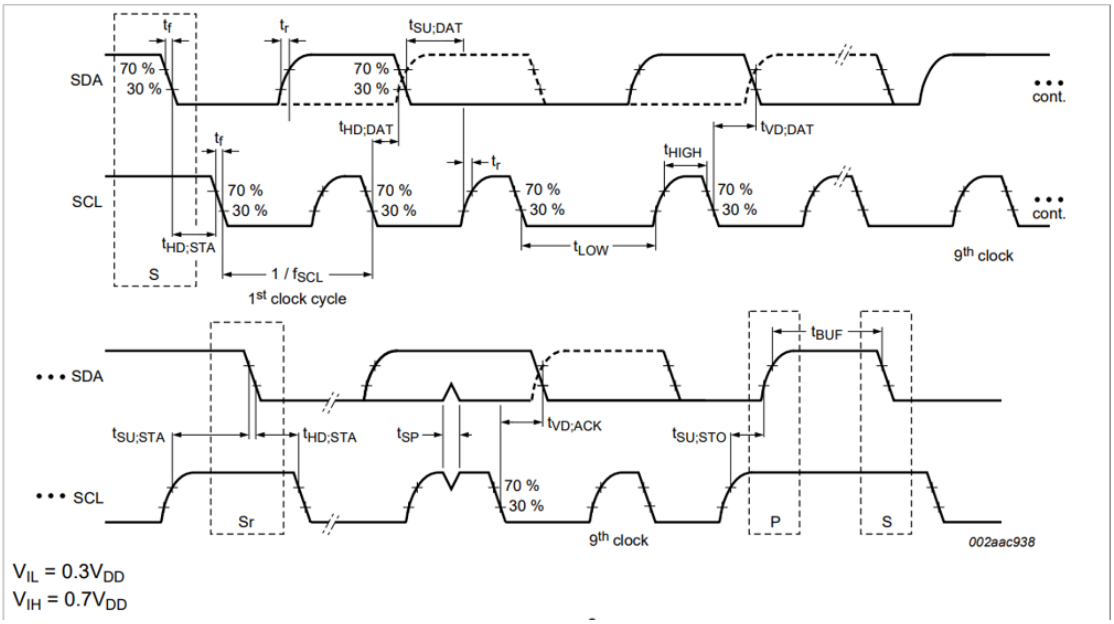

图 6-8：I2C 时序图

类别	说明	CMS8S6990	备注
Fsys	系统时钟频率	48Mhz	
Tsys	系统时钟周期	20.83ns	
TIMER_PRD	I2CMTP[bit 0:6]寄存器值，通过对 TIMER_PRD（进行配置可获得对应的 SCL 时钟频率	0x0~0x7F	TIMER_PRD =0： T_{TIMER_PRD} = Tsys * 3 ; TIMER_PRD !=0： T_{TIMER_PRD}= Tsys * 2*(1+TIMER_PRD)
T_{TIMER_PRD}	内部定时器时钟周期		
VIL	从高到低的翻转电平	2.50V（典型数据）	
VIH	从低到高的翻转电平	2.51V（典型数据）	
F_{SCL}	SCL 时钟频率		
T_{SCL}	SCL 时钟周期		
T_{LOW}	SCL 低电平时间	6 * T_{TIMER_PRD}	
T_{HIGH}	SCL 高电平时间	4 * T_{TIMER_PRD} + 4*Tsys	当 Tsys = 48Mhz: T_{HIGH} =4 * T_{TIMER_PRD} + 4*Tsys 当 Tsys !=48Mhz： T_{HIGH} =4 * T_{TIMER_PRD}
Tr	SDA、SCL 上升时间		根据实际硬件环境测试得到
Tf	SDA、SCL 下降时间		根据实际硬件环境测试得到

综上所述：I2C 模块主控模式 SCL 时钟周期 TSCL=Tf + TLOW + Tr + THIGH =Tf +Tr +10 * TTIMER_PRD。

当 Tsys != 48MHz 时：

1) TIMER_PRD = 0 时，TSCL = Tf + Tr + 10 *3 * Tsys ；

2) TIMER_PRD ！= 0 时，TSCL = Tf + Tr + 10 *2*(1+TIMER_PRD) * Tsys；

6.5.2 应用实例

设置 CMS8S6990 芯片 I2C 模块为主控模式，配置 SCL 时钟的目标值为 400KHz，条件：

1. VDD=5V，Fsys = 48Mhz，Tsys = 20.83ns。
2. SDA、SCL 上拉电阻= 4.7K。
3. VIL = 2.5V，VIH = 2.51V；
   1) 假设 Tf = Tr = 0（理想情况）， 根据公式可以得出 TIMER_PRD = 4.8，故配置 TIMER_PRD = 5；
   2) 开启 I2C 通讯，实测 Tf 与 Tr 的时间，如图 6-9 所示：

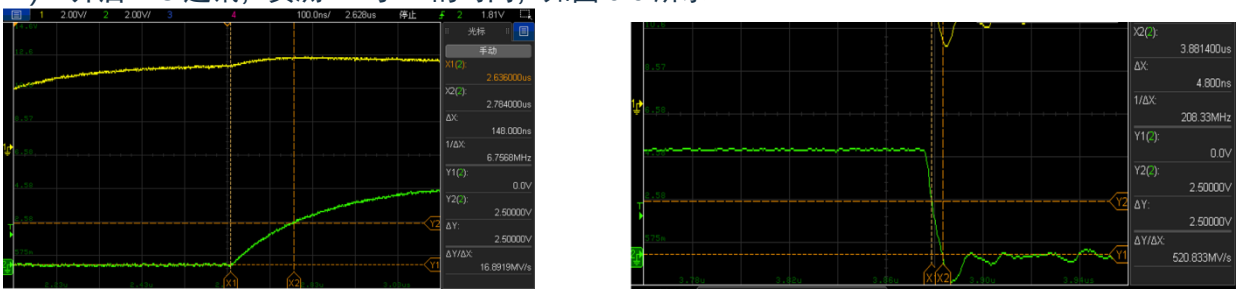

图 6-9：Tf 与 Tr 时间

可以得到 Tr = 148ns； Tf = 4.8ns；

3) 将 Tf、Tr 数据代入公式，得出 TIMER_PRD = 4.43。
   若将 TIMER_PRD 配置为 5；可得出 SCL 频率为 ：366KHz， 实测 SCL 频率为 362.64KHz
   若将 TIMER_PRD 配置为 4；可得出 SCL 频率为 ： 431.2KHz, 实测 SCL 频率为 427KHz；

6.5.3 SCL 频率误差分析

导致 SCL 频率误差的原因主要有以下几点：

1. 系统时钟的误差。
2. VIL、VIH 电平误差导致 获取的 Tr、Tf 时间不准确。
3. 寄存器设置值引入的误差。

7. 更多信息

更多信息，请登录中微半导体网站查看 http://www.mcu.com.cn

8. 版本修订说明

版本号	时间	作者	修改内容
V1.0.0	2022 年 1 月	Pengl	初始版本