

I2C 接口使用指南

版本：1B

<http://wch.cn>

一、I2C 主模式发送

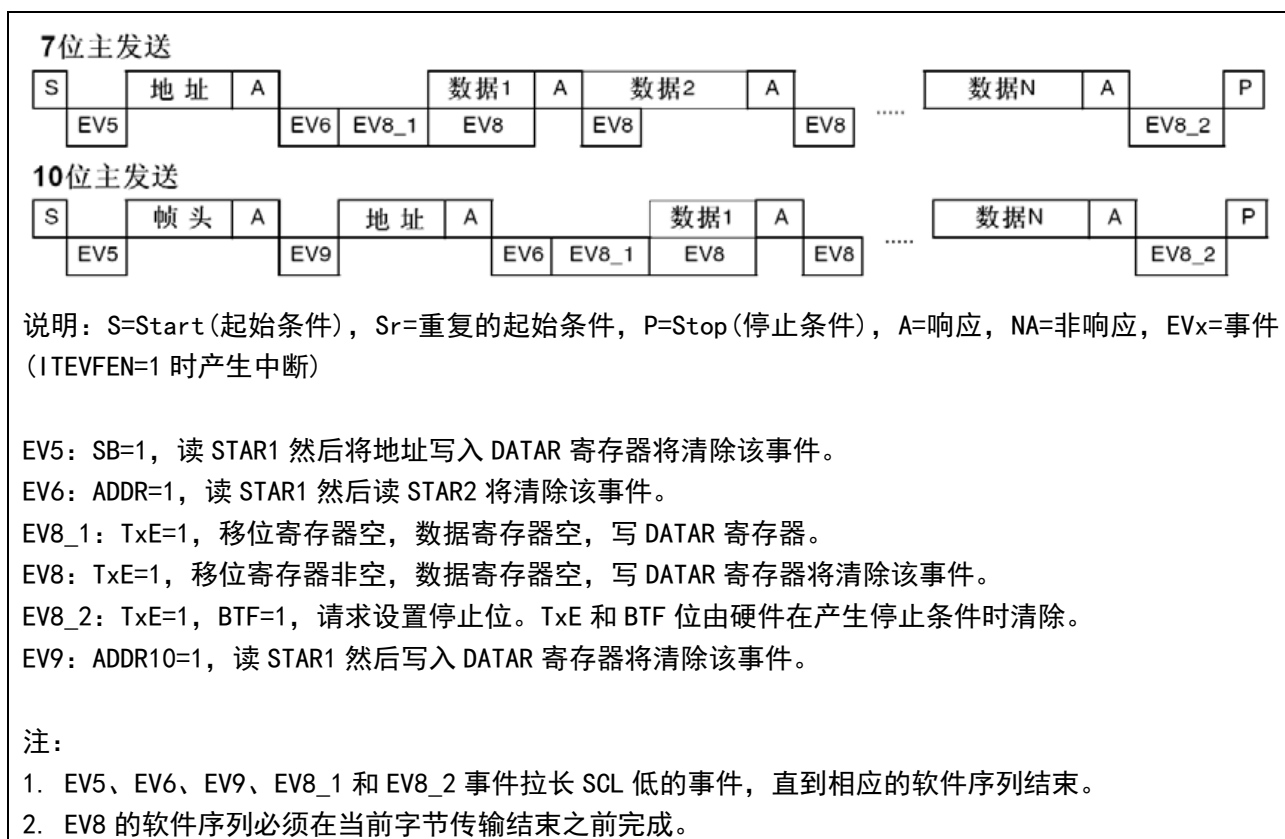


图 1-1 主发送器传送序列图

库函数操作步骤：

1. I2C 硬件初始化

```
void I2C_Init(      I2C_ModeTypeDef    I2C_Mode,
                   UINT32              I2C_ClockSpeed,
                   I2C_DutyTypeDef     I2C_DutyCycle,
                   I2C_AckTypeDef      I2C_Ack,
                   I2C_AckAddrTypeDef  I2C_AckAddr,
                   UINT16              I2C_OwnAddress1 )
```

I2C_Mode	将接口初始化位 I2C 模式: I2C_Mode_I2C
I2C_ClockSpeed	主模式需要提供同步时钟，设置范围为 0~400000bit/s
I2C_DutyCycle	在快速模式下 (100kbits/s~400kbits/s) 需要设置时钟占空比
I2C_Ack	主模式发送用不到 ACK 应答，不需要设置
I2C_AckAddr	主模式不需要设置设备地址的位数
I2C_OwnAddress1	主模式不需要设置设备地址

2. 等待总线处于空闲状态 (BUSY=0)

I2C_GetFlagStatus(I2C_FLAG_BUSY): 获取 BUSY 位的状态

3. I2C 接口发送起始信号 (START=1)

I2C_GenerateSTART(ENABLE): START 位置位

4. 等待总线处于忙状态 (BUSY=1), 接口为主模式 (MSL=1), 起始信号发送完成 (SB=1)

I2C_CheckEvent(I2C_EVENT_MASTER_MODE_SELECT): 获取 BUSY, MSL, SB 的状态

5. 发送 7 位从机地址, 同时将最低位的读写位设置为写状态 (xxxxxxx0)

I2C_Send7bitAddress(RxAddress, I2C_Direction_Transmitter): 往数据寄存器 DATAR 的 7~1 位写入从机地址, 第 0 位置 0 为主模式发送

6. 等待总线处于忙状态 (BUSY=1), 接口为主模式 (MSL=1), 地址发送结束 (ADDR=1), 数据发送完成 (TxE=1), 主机处于发送模式 (TRA=1)

I2C_CheckEvent(I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED): 获取 BUSY, MSL, ADDR, TxE, TRA 的状态

7. 等待数据寄存器为空 (TxE=1)

I2C_GetFlagStatus(I2C_FLAG_TXE): 获取 TxE 的状态

8. 将待发送的字节写入到数据寄存器 DATAR 中

I2C_SendData(Data): 写数据, 如果数据未发送结束, 就跳转到第 7 步

9. 等待总线处于忙状态 (BUSY=1), 接口为主模式 (MSL=1), 主机处于发送模式 (TRA=1), 数据寄存器为空 (TxE=1), 字节发送结束 (BTF=1)

I2C_CheckEvent(I2C_EVENT_MASTER_BYTE_TRANSMITTED): 获取 BUSY, MSL, TRA, TxE, BTF 的状态

10. 发送停止/重起始信号 (STOP=1/START=1)

I2C_GenerateSTOP(ENABLE): STOP 位置位

I2C_GenerateSTART(ENABLE): START 位置位

二、I2C 主模式接收

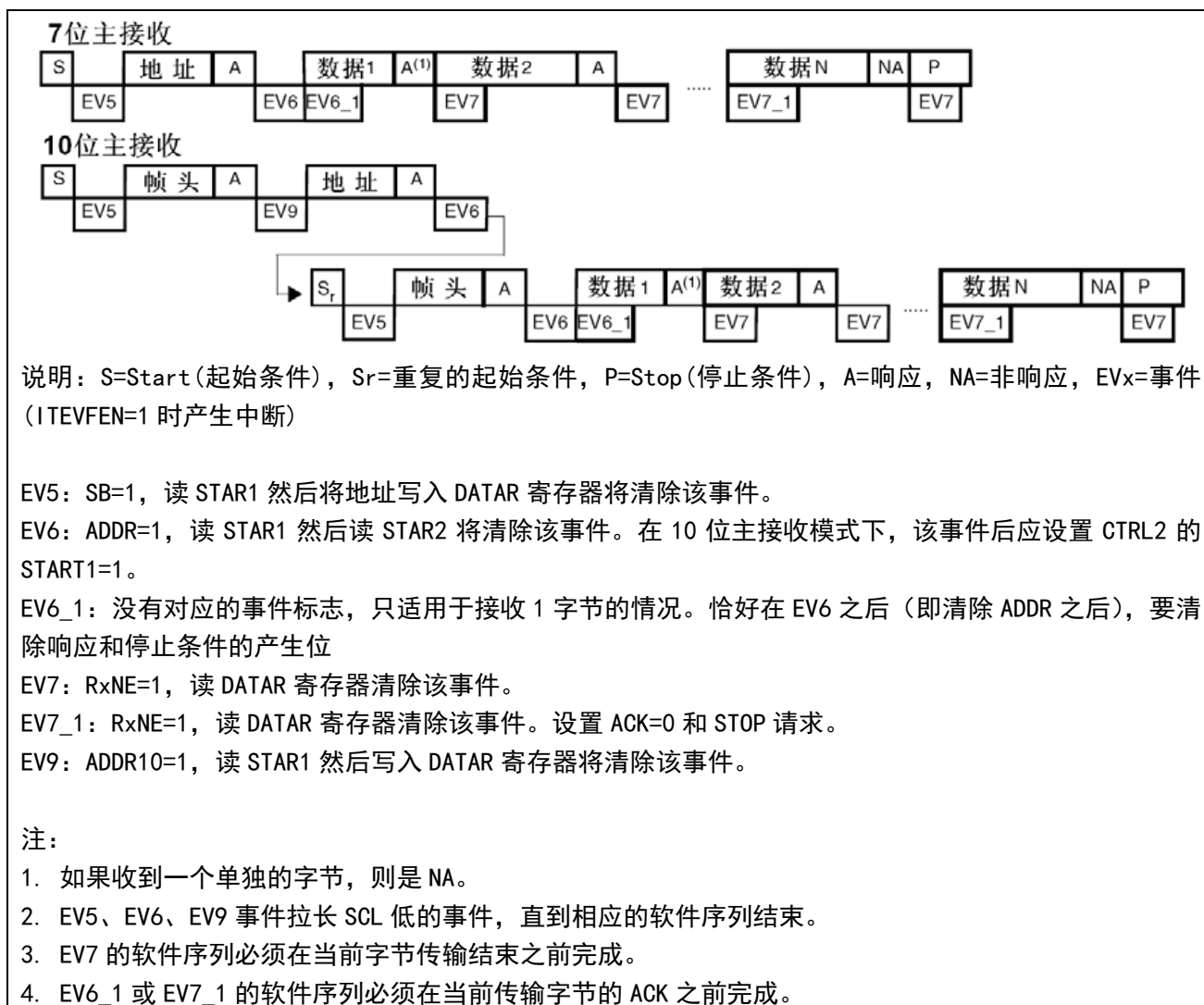


图 2-1 主接收器传送序列图

库函数操作步骤:

1. I2C 硬件初始化

```
void I2C_Init(      I2C_ModeTypeDef    I2C_Mode,
                   UINT32              I2C_ClockSpeed,
                   I2C_DutyTypeDef     I2C_DutyCycle,
                   I2C_AckTypeDef      I2C_Ack,
                   I2C_AckAddrTypeDef  I2C_AckAddr,
                   UINT16              I2C_OwnAddress1 )
```

I2C_Mode	将接口初始化位 I2C 模式: I2C_Mode_I2C
I2C_ClockSpeed	主模式需要提供同步时钟, 设置范围为 0~400000bit/s
I2C_DutyCycle	在快速模式下 (100kbits/s~400kbits/s) 需要设置时钟占空比
I2C_Ack	主模式接收需要设置 ACK=1
I2C_AckAddr	主模式不需要设置设备地址的位数

I2C_OwnAddress1	主模式不需要设置设备地址
-----------------	--------------

2. 等待总线处于空闲状态 (BUSY=0)

I2C_GetFlagStatus(I2C_FLAG_BUSY): 获取 BUSY 位的状态

3. I2C 接口发送起始信号 (START=1)

I2C_GenerateSTART(ENABLE): START 位置位

4. 等待总线处于忙状态 (BUSY=1), 接口为主模式 (MSL=1), 起始信号发送完成 (SB=1)

I2C_CheckEvent(I2C_EVENT_MASTER_MODE_SELECT): 获取 BUSY, MSL, SB 的状态

5. 发送 7 位从机地址, 同时将最低位的读写位设置为读状态 (xxxxxx1)

I2C_Send7bitAddress(RxAddress, I2C_Direction_Receiver): 往数据寄存器 DATAR 的 7~1 位写入从机地址, 第 0 位置 1 为主模式接收

6. 等待总线处于忙状态 (BUSY=1), 接口为主模式 (MSL=1), 地址发送结束 (ADDR=1)

I2C_CheckEvent(I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED): 获取 BUSY, MSL, ADDR 的状态

7. 等待数据寄存器非空 (RxNE=1)

I2C_GetFlagStatus(I2C_FLAG_RXNE): 获取 RxNE 的状态

8. 读取数据寄存器 DATAR 中接收到的数据

Data=I2C_ReceiveData(): 读数据, 如果倒数第 2 个数据未接收结束, 就跳转到第 7 步

9. 主设备为了能在收到最后一个字节后产生一个 NACK 脉冲, 必须在读取倒数第二个字节之后 (倒数第二个 RxNE 事件之后) 清除 ACK 位 (ACK=0)

I2C_AcknowledgeConfig(DISABLE): 清除 ACK 位

10. 主设备为了能在收到最后一个字节后产生一个停止/重起始信号, 必须在读取倒数第二个字节之后 (倒数第二个 RxNE 事件之后) 设置 STOP/START 位 (STOP=1/START=1)

I2C_GenerateSTOP(ENABLE): STOP 位置位

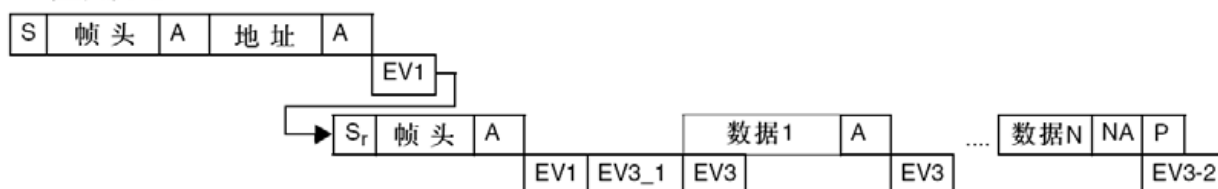
I2C_GenerateSTART(ENABLE): START 位置位

三、I2C 从模式发送

7位从发送



10位从发送



说明：S=Start(起始条件)，Sr=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，EVx=事件 (ITEVFEN=1 时产生中断)

EV1: ADDR=1, 读 STAR1 然后读 STAR2 将清除该事件。

EV3_1: Tx=1, 移位寄存器空, 数据寄存器空, 写 DATAR。

EV3: Tx=1, 移位寄存器非空, 数据寄存器空, 写 DATAR 将清除该事件。

EV3_2: AF=1, 在 STAR1 寄存器的 AF 位写 '0' 可清除 AF 位。

注：

1. EV1、EV3_1 事件拉长 SCL 低的事件，直到相应的软件序列结束。
2. EV3 的软件序列必须在当前字节传输结束之前完成。

图 3-1 从发送器的传送序列图

库函数操作步骤：

1. I2C 硬件初始化

```
void I2C_Init(      I2C_ModeTypeDef      I2C_Mode,
                   UINT32                  I2C_ClockSpeed,
                   I2C_DutyTypeDef         I2C_DutyCycle,
                   I2C_AckTypeDef          I2C_Ack,
                   I2C_AckAddrTypeDef      I2C_AckAddr,
                   UINT16                  I2C_OwnAddress1 )
```

I2C_Mode	将接口初始化位 I2C 模式: I2C_Mode_I2C
I2C_ClockSpeed	从模式不需要设置同步时钟
I2C_DutyCycle	从模式不需要设置时钟占空比
I2C_Ack	从模式发送需要设置 ACK=1
I2C_AckAddr	从模式需要设置设备地址的位数
I2C_OwnAddress1	从模式需要设置设备地址

2. 等待总线处于忙状态 (BUSY=1)，收到的地址匹配 (ADDR=1)

I2C_CheckEvent(I2C_EVENT_SLAVE_RECEIVER_ADDRESS_MATCHED)：获取 BUSY，ADDR 的状态

3. 等待数据寄存器为空 (Tx=1)

I2C_GetFlagStatus(I2C_FLAG_TXE)：获取 Tx=1 的状态

4. 将待发送的字节写入到数据寄存器 DATAR 中

I2C_SendData(Data): 写数据, 如果数据未发送结束, 就跳转到第 3 步

5. 等待主机应答 NACK(AF=1)

I2C_CheckEvent(I2C_EVENT_SLAVE_ACK_FAILURE): 获取 AF 的状态

四、I2C 从模式接收

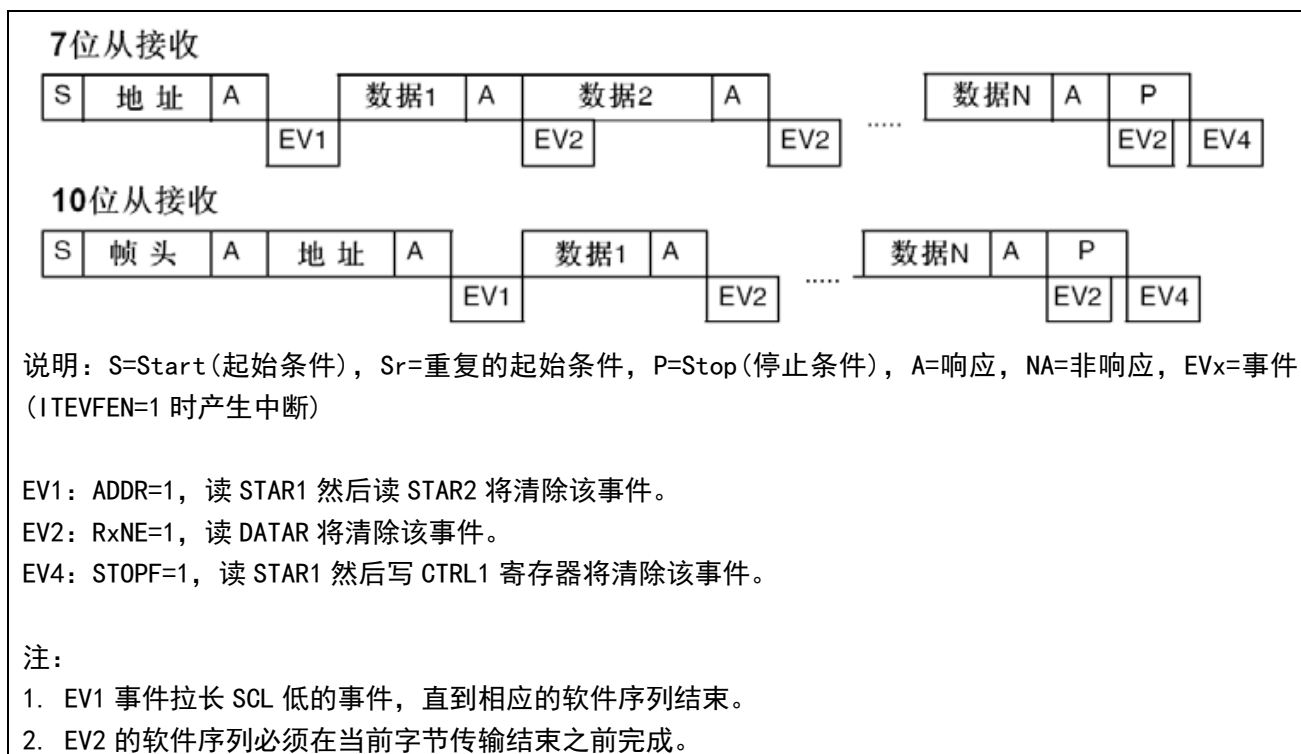


图 4-1 从接收器的传送序列图

库函数操作步骤：

1. I2C 硬件初始化

```
void I2C_Init(      I2C_ModeTypeDef      I2C_Mode,
                   UINT32                  I2C_ClockSpeed,
                   I2C_DutyCycleTypeDef    I2C_DutyCycle,
                   I2C_AckTypeDef          I2C_Ack,
                   I2C_AckAddrTypeDef       I2C_AckAddr,
                   UINT16                   I2C_OwnAddress1 )
```

I2C_Mode	将接口初始化位 I2C 模式: I2C_Mode_I2C
I2C_ClockSpeed	从模式不需要设置同步时钟
I2C_DutyCycle	从模式不需要设置时钟占空比
I2C_Ack	从模式接收需要设置 ACK=1
I2C_AckAddr	从模式需要设置设备地址的位数
I2C_OwnAddress1	从模式需要设置设备地址

2. 等待总线处于忙状态 (BUSY=1)，收到的地址匹配 (ADDR=1)

I2C_CheckEvent(I2C_EVENT_SLAVE_RECEIVER_ADDRESS_MATCHED)：获取 BUSY，ADDR 的状态

3. 等待数据寄存器非空 (RxNE=1)

I2C_GetFlagStatus(I2C_FLAG_RXNE)：获取 RxNE 的状态

4. 读取数据寄存器 DATAR 中接收到的数据

Data=I2C_ReceiveData()：读数据，如果数据未接收结束，就跳转到第 3 步

5. 等待主机发送 STOP 信号 (STOPF=1)

I2C_CheckEvent(I2C_EVENT_SLAVE_STOP_DETECTED)：获取 STOPF 的状态