IIC模块通解

由 enlai.feng创建于二月 21, 2024

1.基本介绍

I2C，全名Inter-Integrated Circuit（互联集成电路），是一种用于在微控制器、传感器、外围设备等之间传输数据的串行通信协议和物理接口标准。I2C协议允许多个设备通过共享同一对数据线（SDA）和时钟线（SCL）来进行通信，这使得它在连接多个设备时非常有用，尤其是在嵌入式系统和电子设备中。

2.硬件结构

I2C共有两种类型的设备：**主设备**和**从设备**

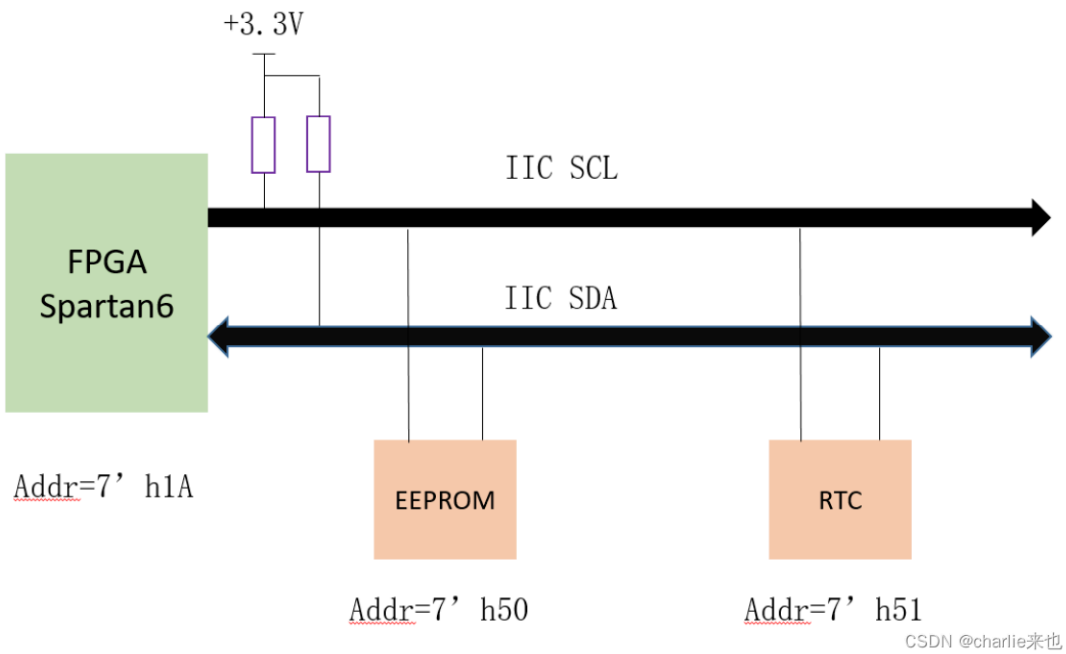
主设备 - 初始化传输（START命令）、生成时钟（SCL）信号并终止传输（STOP命令）的组件。主设备可以是发送器或接收器。

从设备 - 主设备寻址的设备。从设备可以是接收器或发送器。

（1）IIC工作模式

I2C 总线支持多主和主从两种工作方式，通常工作在主从工作方式。在主从工作方式中，主机启动数据的发送（发出启动信号）并产生时钟信号，数据发送完成后，发出停止信号。

（2）总线结构



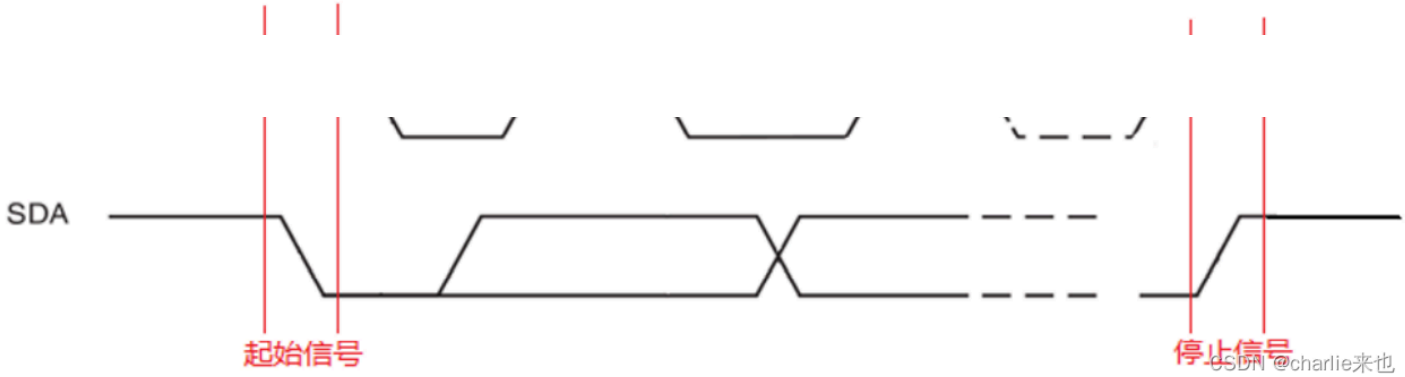
I2C 总线由数据线 SDA 和时钟线 SCL 构成通信线路，既可用于发送数据，也可接收数据。在主控与被控之间可进行双向数据传送，各种被控器件均并联在总线上，通过器件地址（SLAVE ADDR，具体可查器件手册）识别。

图中的 I2C_SCL 是串行时钟线，I2C_SDA 是串行数据线，由于 I2C 器件一般采用开漏结构与总线相连，所以 I2C_SCL 和 I2C_SDA 均需接上拉电阻，也正因此，当总线空闲时，这两条线路都处于高电平状态，当连到总线上的任一器件输出低电平，都将使总线拉低，即各器件的 SDA 及 SCL 都是“线与”关系。**总线上的设备数量最大限制仅受最大电容规范的限制。**

（3）通信过程

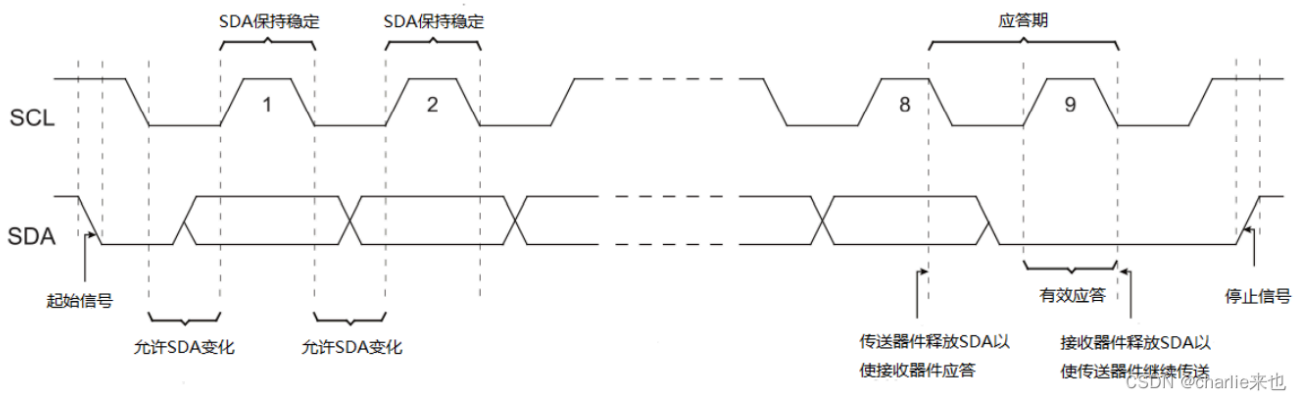
空闲状态：SCL和SDA均处于高电平

主机开始数据传输：在 SCL 为高电平时将 SDA 线拉低，产生一个起始信号，从机检测到起始信号后，准备接收数据，当数据传输完成，主机需产生一个停止信号，告诉从机数据传输结束，停止信号的产生是在 SCL 为高电平时，SDA 从低电平跳变到高电平，从机检测到停止信号后，停止接收数据。



I2C 整体时序如上图。起始信号之前为空闲状态，起始信号之后到停止信号之前的这一段为数据传输状态，主机可以向从机写数据，也可以读取从机输出的数据，数据的传输由双向数据线（SDA）完成。停止信号产生后，总线再次处于空闲状态。

3.数据传输格式



在起始信号之后，主机开始发送传输的数据；
在SCL 为低电平状态时，SDA 允许改变传输的数据位（1 为高电平，0 为低电平），
在SCL 为高电平状态时，SDA 要求保持稳定，
相当于一个时钟周期传输 1bit 数据，经过 8 个时钟周期后，传输了 8bit 数据，即一个字节。

从机应答成功和失败：
第 8 个时钟周期末，主机释放 SDA 以使从机应答，在第 9 个时钟周期，**从机将 SDA 拉低以应答**——既应答成功；

如果第 9 个时钟周期，SCL 为高电平时，SDA 未被检测到为低电平，视为非应答，表明此次数据传输失败。

应答之后：
第 9 个时钟周期末，从机释放 SDA 以使主机继续传输数据，如果主机发送停止信号，此次传输结束。

注：数据以 8bit 即一个字节为单位串行发出，其最先发送的是字节的最高位。

4.设备地址

每个 I2C 器件都有一个器件地址，有些 I2C 器件的器件地址是固定的(可以查阅器件手册)，而有些 I2C 器件的器件地址由一个固定部分和一个可编程的部分构成，如对于一个EEPROM，AT24C64来说其器件地址为 1010 加 3 位的可编程地址，3 位可编程地址由器件上的 3 个管脚A2、A1、A0的硬件连接决定。当硬件电路上分别将这三个管脚连接到 GND 或 VCC 时，就可以设置不同的可编程地址。

I2C支持7bit和10bit两种格式的设备地址。

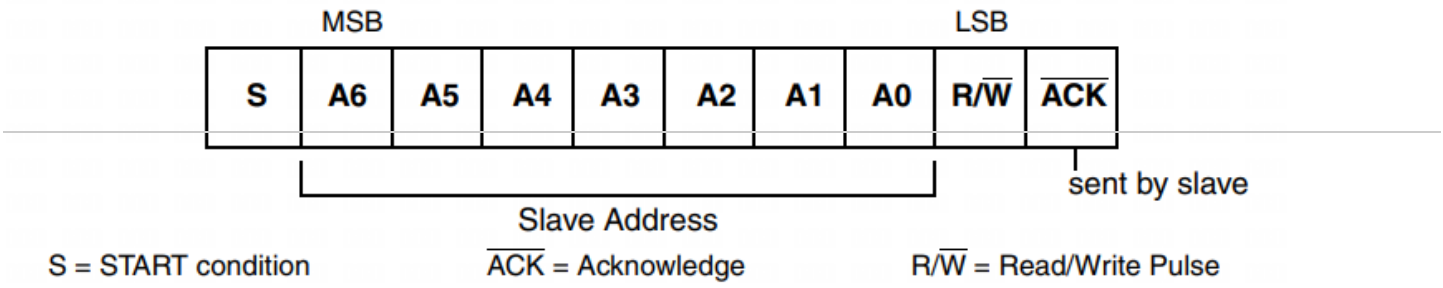
IIC数据传输速率：

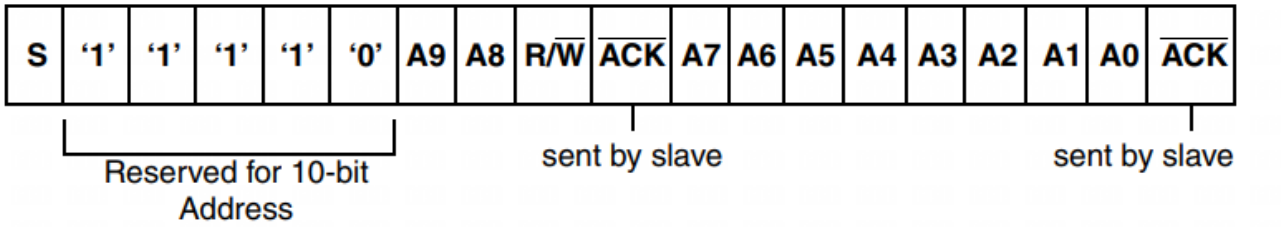
标准模式下可达 100kbit/s
快速模式下可达 400kbit/s
高速模式下可达 3.4Mbit/s

5.软件结构

传输过程

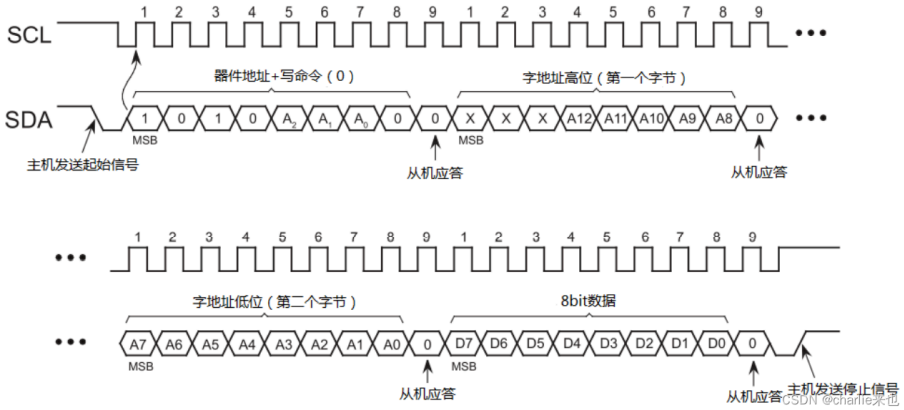
设备地址：进行数据传输时，主机首先向总线上发出开始信号，对应开始位 S，然后按照从高到低的位序发送器件地址，一般为 7bit，第 8bit 位为读写控制位 R/W，该位为 0 时表示主机对从机进行写操作，当该位为 1 时表示主机对从机进行读操作，然后接收从机响应。如果器件地址为10位地址，则分为两个字节传输（先传高位）。



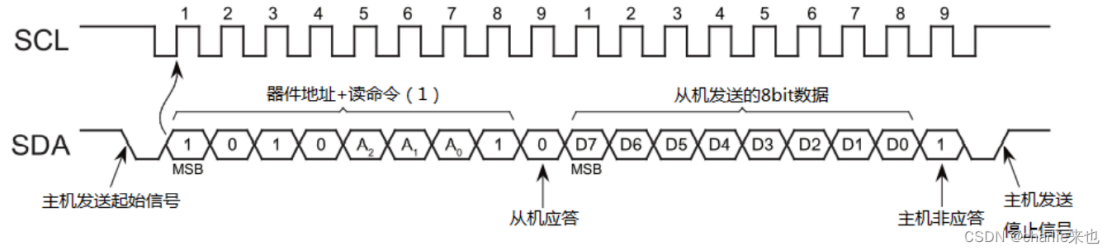


字地址：当我们对一个器件中的存储单元（包括寄存器）进行读写时，首先要指定存储单元的地址即字地址，然后再向该地址写入内容。该地址为一个或两个字节长度，具体长度由器件内部的存储单元的数量决定。

写命令：



读命令：



6. IIC-tools工具

（1）i2cdetect

i2cdetect用來列举I2C bus和上面所有的设备，可接受的参数有：

```
Usage: i2cdetect [-y] [-a] [-q|-r] I2CBUS [FIRST LAST]
       i2cdetect -F I2CBUS
       i2cdetect -l
  I2CBUS is an integer or an I2C bus name
  If provided, FIRST and LAST limit the probing range.
```

-V：输出当前版本号。

```
root@Linux:/# i2cdetect -V
i2cdetect version 4.1
```

-l：输出所有 i2c 总线，如下总线编号有twi1和twi2，或者1和2。

```
                                                       I2C adapter
i2c-2   i2c             twi2                           I2C adapter
root@Linux:/# ls -l /dev/i2c-*
crw-------    1 root     root       89,   1 Jan  1 23:11 /dev/i2c-1
crw-------    1 root     root       89,   2 Jan  1 23:11 /dev/i2c-2
```

I2CBUS：i2c总线编号
-F：此 i2c 支持的功能：

```
root@Linux:/# i2cdetect -F 1
Functionalities implemented by /dev/i2c-1:
I2C                              yes
SMBus Quick Command              yes
SMBus Send Byte                  yes
SMBus Receive Byte               yes
SMBus Write Byte                 yes
SMBus Read Byte                  yes
SMBus Write Word                 yes
SMBus Read Word                  yes
SMBus Process Call               yes
SMBus Block Write                yes
SMBus Block Read                 no
SMBus Block Process Call         no
SMBus PEC                        yes
I2C Block Write                  yes
I2C Block Read                   yes
```

(2) i2cdump

i2cdump读取设备上所有寄存器的值，可接受的参数有：

```
Usage: i2cdump [-f] [-y] [-r first-last] [-a] I2CBUS ADDRESS [MODE [BANK [BANKREG]]]
  I2CBUS is an integer or an I2C bus name
  ADDRESS is an integer (0x03 - 0x77, or 0x00 - 0x7f if -a is given)
  MODE is one of:
    b (byte, default)
    w (word)
    W (word on even register addresses)
    s (SMBus block)
    i (I2C block)
    c (consecutive byte)
    Append p for SMBus PEC
```

-V：输出当前版本号
-f：强制使用此设备地址，即使此设备地址已经被使用；若不添加此参数，地址可能写失败
-y：指令执行自动 yes，否则会提示确认执行Continue? [Y/n] Y，不加参数y会有很多执行提示，可以帮助判断
-r：读取从 first-last 之间的寄存器值
-a：读取0x00-0xff范围的地址

(3) i2cset

i2cset设置设备上寄存器的值，可接受的参数有：

```
Usage: i2cset [-f] [-y] [-m MASK] [-r] [-a] I2CBUS CHIP-ADDRESS DATA-ADDRESS [VALUE] ... [MODE]
  I2CBUS is an integer or an I2C bus name
  ADDRESS is an integer (0x03 - 0x77, or 0x00 - 0x7f if -a is given)
  MODE is one of:
    c (byte, no value)
    b (byte data, default)
```

```
    w (word data)
    i (I2C block data)
```

-V：输出当前版本号
-f：强制使用此设备地址，即使此设备地址已经被使用；若不添加此参数，地址可能写失败
-y：指令执行自动 yes，否则会提示确认执行Continue? [Y/n] Y，不加参数y会有很多执行提示，可以帮助判断
-m：添加掩码
-r：回显，显示是否写入成功，要写的值和读取的值
-a：允许使用0x00-0x02和0x78-0x7f之间的地址
I2CBUS：i2c总线编号
CHIP-ADDRESS：设备地址
DATA-ADDRESS：要写入的寄存器地址
VALUE：要写入的值
MODE：数据长度类型

(4) i2cget

i2cget读取设备上寄存器的值，可接受的参数有:

```
Usage: i2cget [-f] [-y] [-a] I2CBUS CHIP-ADDRESS [DATA-ADDRESS [MODE]]
  I2CBUS is an integer or an I2C bus name
  ADDRESS is an integer (0x03 - 0x77, or 0x00 - 0x7f if -a is given)
  MODE is one of:
    b (read byte data, default)
    w (read word data)
    c (write byte/read byte)
    Append p for SMBus PEC
```

-V：输出当前版本号
-f：强制使用此设备地址，即使此设备地址已经被使用；若不添加此参数，地址可能写失败
-y：指令执行自动 yes，否则会提示确认执行Continue? [Y/n] Y，不加参数y会有很多执行提示，可以帮助判断
-a：允许使用0x00-0x02和0x78-0x7f之间的地址
I2CBUS：i2c总线编号
CHIP-ADDRESS：设备地址
DATA-ADDRESS：要读取的寄存器地址
MODE：数据长度类型

```
//0x38设备地址，0x04要读取的寄存器
root@Linux:/# i2cget -f -y 1 0x38 0x04
0x45
```

(5) i2ctransfer

i2ctransfer通过一次传输发送用户定义的I2C消息，用于创建I2C消息并将其作为一次传输合并发送。对于已读消息，已接收缓冲区的内容被打印到stdout，每条已读消息一行.

```
Usage: i2ctransfer [-f] [-y] [-v] [-V] [-a] I2CBUS DESC [DATA] [DESC [DATA]]...
  I2CBUS is an integer or an I2C bus name
  DESC describes the transfer in the form: {r|w}LENGTH[@address]
    1) read/write-flag 2) LENGTH (range 0-65535) 3) I2C address (use last one if omitted)
  DATA are LENGTH bytes for a write message. They can be shortened by a suffix:
    = (keep value constant until LENGTH)
    + (increase value by 1 until LENGTH)
    - (decrease value by 1 until LENGTH)
    p (use pseudo random generator until LENGTH with value as seed)

Example (bus 0, read 8 byte at offset 0x64 from EEPROM at 0x50):
  # i2ctransfer 0 w1@0x50 0x64 r8
Example (same EEPROM, at offset 0x42 write 0xff 0xfe ... 0xf0):
  # i2ctransfer 0 w17@0x50 0x42 0xff-
```

-V：输出当前版本号

-f：强制使用此设备地址，即使此设备地址已经被使用；若不添加此参数，地址可能写失败

-y：指令执行自动 yes，否则会提示确认执行Continue? [Y/n] Y，不加参数y会有很多执行提示，可以帮助判断
-v：启用详细输出
-a：允许使用0x00-0x02和0x78-0x7f之间的地址

{r|w}指定消息是读还是写
<消息长度>指定在此消息中读取或写入的字节数。它被解析为一个无符号的16位整数
[@设备地址]指定此消息要访问的芯片的7位地址，并且是整数。如果省略，请重用以前的地址。通常，将阻止0x03-0x77范围之外的地址以及附加了内核驱动程序的地址。通过-f（强制），可以使用所有地址
如果I2C消息是写操作，则随后是带有要写数据的数据块。它由<消息长度> 个字节组成，这些字节可以用十六进制，八进制等的常用前缀进行标记。为了更轻松地轻松创建较大的数据块，该数据字节可以带有一个后缀。