单片机 C51 实现 I2C 虚拟总线

The simulating application of I2C-BUS by 80C51

摘要：本文介绍了 I2C 总线的基本结构、工作原理，并使用单片机 C51 编程实现 I2C 总线的功能，即 I2C 总线的虚拟技术。

关键字：C51; I2C; 总线;
分类号：TN911
文献标识码：A

Abstract: The basic structure and operation criterion of I2C-BUS were introduced in this paper, and the functions of I2C-BUS were realized by 80C51, this is the simulated technology of I2C-BUS.

Keywords: C51; I2C; bus;

1．前言

I2C(Inter－Integrated Circuit)总线是由 Philips 公司推出的一种二线制串行总线，用于连接微控制器及其外围设备。它允许若干兼容器件(如存储器、A/D 和 D/A 转换器，以及 LED、LCD 驱动器等)共享总线。它是同步通信的一种特殊形式，具有接口线路少，控制方式简便，器件封装体积小，通信速率较高等优点。近年来 I2C 在微电子通信控制等领域得到广泛的应用。

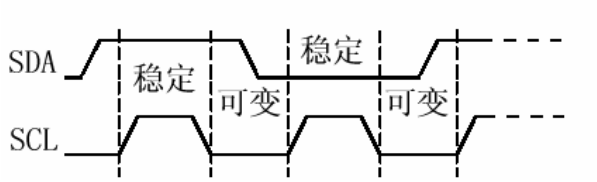
2．I2C 总线概述

I2C 总线通过串行数据线(SDA)和串行时钟线(SCL)与连到总线上的IC器件之间进行数据传输。SDA 和 SCL 都是双向 I/O 线路，通过上拉电阻连接正电源，连接总线的器件的输出级必须是集电极开路或漏电极开路，这样才能够实现“线与”功能。在标准模式下，I2C 总线的传输速率可达 100Kbps。

I2C 总线是一种多主机的总线，即可以连接多个能控制总线的器件，但同一时刻只能有一个器件控制总线而成为主机。

3．位传输

I2C 总线空闲时，SDA 和 SCL 都保持高电平。数据传输时，每传输一个数据位必须产生一个时钟脉冲，时钟脉冲一般由主机产生。SDA 线上的数据必须在 SCL 高电平周期保持稳定，SDA 线的高低电平状态只能在SCL线是低电平时才能改变(如图一)。在标准模式下，SCL 线高低电平宽度必须大于 4.7us。



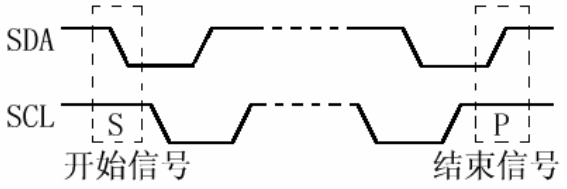
图一　位传输

4．开始信号和结束信号

开始信号(S)和结束信号(P)由主机产生。总线在开始信号以后被认为处于忙的状态，在结束信号以后被认为再次处于空闲状态。在传输过程中，由于传输错误等原因，可以再次产生一个开始信号，即重复开始信号。

●开始信号(重复开始信号)：SCL 为高电平时，SDA 由高电平向低电平跳变，开始传送数据。

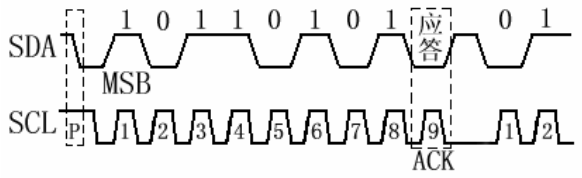
●结束信号：SCL 为低电平时，SDA 由低电平向高电平跳变，结束传送数据。(如图二)。



图二　开始和结束信号

5．数据传输

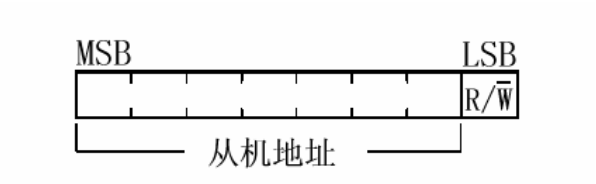
开始信号后，就可以进行数据的传输。主机发送到SDA线上的每个字节必须为8位，首先传输的是数据的最高位(MSB)。每次传输可以发送的字节数量不受限制。主机或从机每接收一个字节后必须返回一个响应位(ACK)。(如图三)。



图三　数据传输

6．寻址字节

主机启动开始信号以后，发送一个寻址字节，该字节的高 7 位为从机地址，最低位(LSB)为数据方向位，“0”表示写，“1”表示读。(如图四)。从机地址由一个固定的部分和一个可编程的部分组成，一般为高 4 位的固定地址和低 3 位的可编程地址。I2C 总线委员会协调 I2C 地址的分配。



图四　寻址字节

7．程序实现

在实际的 I2C 总线应用中，经常采用以一个单片机为主机，其他接口器件为从机的单主机结构。目前有相当多的单片机提供支持 I2C 总线的接口，不使用单片机的 I2C 接口，直接通过 I/O 线编程来实现 I2C 总线的功能，即为 I2C 总线的虚拟技术。以下提供了虚拟 I2C 总线的 C51 源代码，并对程序进行详细的注释。

```
#include <intrins.h>
#define uchar unsigned char
#define Delay1us() _nop_()
#define Delay2us() _nop_();_nop_()
#define Delay5us() _nop_();_nop_();
            _nop_();_nop_();_nop_()

sbit SDA=P1^0;       //模拟数据线
sbit SCL=P1^1;       //模拟时钟线

//开始信号
void START_I2C(void)
{
  SDA=1;
  SCL=1;             //空闲状态
  Delay5us();
  SDA=0;             // 1->0 跳变，开始
  Delay5us();
  SCL=0;             //准备发送数据
  Delay1us();
}

//结束信号
void STOP_I2C(void)
{
  SDA=0;
  SCL=1;
  Delay5us();
  SDA=1;             // 0->1 跳变，结束
  Delay2us();
}

//应答信号
void ACK_I2C(bit ack)
{
  SDA=ack?1:0;       //发出应答或非应答
  Delay2us();
  SCL=1;             //保持一个高电平周期
  Delay5us();
  SCL=0;
  Delay2us();
}

//发送一个字节
bit SEND_BYTE(uchar byte)
{
  uchar i;
  bit ack;
  for(i=0;i<8;i++)
  {
    SDA=((byte<<i)&0x80)?1:0; //赋值发送位
    Delay1us();
    SCL=1;           //保持一个高电平周期
    Delay5us();
    SCL=0;
  }
  Delay2us();
  SDA=1;             //准备接收应答位
  Delay2us();
  SCL=1;
  Delay2us();
  ack=SDA?1:0;       //判断应答位
  SCL=0;
  return(ack);       //返回应答位
}
```

```
//接收一个字节
uchar RECEIVE_BYTE(void)
{
  uchar i;
  uchar byte=0;
  for(i=0;i<8;i++)
  {
    SDA=1;          //输入方式，等待接收
    SCL=0;          //提供一个低电平周期
    Delay5us();
    SCL=1;
    Delay2us();
    byte=byte<<1;
    if(SDA)         //读数据位，存入 byte
      byte+=1;
  }
  SCL=0;
  Delay2us();
  return(byte);     //返回接收字节
}

//向从机写一个字节
bit WRITE_BYTE(uchar address, uchar byte)
{
  START_I2C();
  if(SEND_BYTE(address))
    return 1;       //非应答，返回
  if(SEND_BYTE(byte))
    return 1;       //非应答，返回
  STOP_I2C();
  return 0;
}

//读取从机一个字节
uchar READ_BYTE(uchar address, uchar *byte)
{
  START_I2C();
  if(SEND_BYTE(address))
    return 1;       //非应答，返回
  *byte=RECEIVE_BYTE();
  ACK_I2C(0);       //应答信号
  STOP_I2C();
  return 0;
}
```

参考文献

[1] THE I2C-BUS SPECIFICATION
VERSION 2.1 JANUARY 2000

[2] 周立功等，增强型 80C51 单片机速成与实战。北京航空航天大学出版社，2003 年