doi:10.16018/j.cnki.cn32-1650/n.201703010

MSP430 单片机 IIC 总线通信控制系统研究

杨汉华
(盐城工学院 学生处,江苏 盐城　224002)

摘要:为了解决 IIC 总线通信过程中经常出现的不同问题,充分利用 MSP430 单片机寄存器设置和中断设置等的特点和优良性,采用硬件设计和软件调试相结合的方法对存储器电路、键盘电路和显示电路进行了分析和设计,分别采用模拟 IIC 时序方法、寄存器方法和中断方法进行主机和从机的 IIC 通信,并对出现的各种问题提出新的解决方法,然后通过实验进行验证,结果表明基于 MSP430 单片机的 IIC 总线通信设计完全符合设计要求。

关键词:微处理器;逻辑判断;参数变化;IIC 通信
中图分类号:TM341　　**文献标识码**:A　　**文章编号**:1671-5322(2017)03-0048-06

MSP430 系列单片机是美国德州仪器(TI)于 1996 年推向市场的一种功耗较小、具有精简指令集(RISC)的 16 位超低 MSP430 单片机,多应用于需要电池供电的便携式仪器仪表中。该单片机能够将多个不同功能的模拟电路、数字电路模块和微处理器集成在一个芯片上,以提供“单片机”解决方案。MSP430 单片机有很多类型,主要包括 MSP430 F167、MSP430 F168、MSP430 F169、MSP430 F1610 等类别,本文以 MSP430 F169 为例进行 IIC 通信设计。

MSP430 单片机具有 51 系列单片机所不具备的功能,如 IIC 总线功能。由于 MSP430 单片机对 IIC 通信有相应的寄存器设置,通过寄存器设置的 IIC 通信,具有程序简洁、通信速度快等优点,但按照文献[1]给出的设计方案,在 IIC 总线通信过程中总会出现不同的问题。为克服这个缺点,并对 IIC 通信方式的 3 种方法进行比较,本文采用硬件设计和软件调试相结合的方法,充分利用 MSP430 单片机寄存器设置和中断设置等的特点和优良性,分别采用模拟 IIC 时序方法、寄存器方法和中断方法进行主机和从机的 IIC 通信,并对出现的各种问题提出新的解决方法,从而实现基于 MSP430 单片机的 IIC 总线通信的完整设计。具体地说,即以 MSP430 单片机为控制核心,以具有 IIC 总线的芯片 AT24C02 为被控对象,用矩阵式键盘设定数据,通过 IIC 总线把设定数据写入存储器 AT24C02,通过数码管来显示,在断电后又重新上电时,数码管显示的数据保持不变。试验结果表明,该控制方法具有较高的实用价值。

1　MSP430 单片机 IIC 通信的工作原理

IIC 通信协议是一种高性能芯片间的串行同步传输协议。在该协议下,只需 2 根信号线,即 1 根时钟线 SCL、1 根数据线 SDA,就能实现双工同步数据传输,并极其方便地将 128 个不同的设备互连到一起,构成多机系统和外围器件扩展系统。IIC 接口采用器件地址的硬件设置方法,通过软件选址完全避免了器件片选寻址的缺点,使得硬件系统具有简单、灵活的扩展方法[2]。外部硬件只需要 2 个上拉电阻,每根线上 1 个。所有连接到总线上的设备都有自己的地址。

IIC 模块的传输模式为主从式。对系统中的某一器件来说,有 4 种可能的工作方式,分别是:主机发送方式、从机发送方式、主机接收方式和从机接收方式[3]。

2　MSP430 单片机 IIC 通信硬件电路设计

2.1　IIC 通信电路的设计

IIC 总线上所有器件要依靠 SDA 发送的地址

收稿日期:2017-06-29

作者简介:杨汉华(1973—),男,江苏盐城人,副教授,硕士,主要研究方向电子信息工程。

信号寻址,不需要片选线。IIC 器件在出厂时已经给定了地址编码[4],器件地址 SLA 格式如图 1 所示。

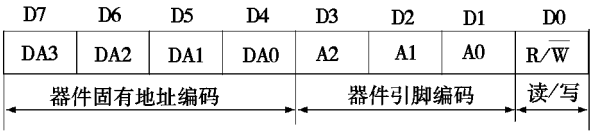


图 1　I^2C 总线器件地址 SLA 格式

Fig. 1　LA format of I2C bus components address

1)DA3 ~ DA0 4 位器件地址是 IIC 总线器件固有地址编码,由出厂时给定,用户不能自行设置。对于 AT24C02 芯片,其芯片地址为 1010,是固定不变的,而 A2A1A0 是变化的。

2)A2A1A0 3 位引脚地址用于相同地址器件的识别。若 IIC 总线上挂有相同地址的器件,或同时挂有多片相同器件时,可用硬件连接方式对 3 位引脚接 VCC 或接地,以形成地址数据。如果 A2A1A0 都是接地,那么 A2A1A0 就是 000。

3)R/W(——)确定数据传送方向:R/W(——)为 1 时,主机接收;R/W(——)为 0 时,主机发送。

如果单片机自带 IIC 总线接口,则所有 IIC 器件对应连接到该总线上即可;若无 IIC 总线接口,则可以使用 I/O 口模拟 IIC 总线。IIC 总线一次完整的数据传输通信格式如图 2 所示。

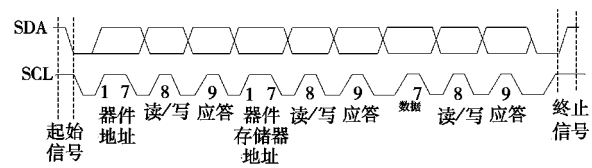


图 2　IIC 总线一次完整的数据传输通信格式

Fig. 2　The format of IIC bus once complete data transmission

数据传输过程中发送到 SDA 线上的每个字节必须是 8bit,每次传输可以发送的字节数量不受限制,但每个字节后必须跟一个响应位,传输数据时高位在前。

时钟 SCL 为高电平期间,SDA 线上的数据必须保持稳定,只有在 SCL 为低电平时 SDA 线上的数据状态才允许改变。本设计由于只采用一个 AT24C02 存储器,因此把此芯片地址线全部接地,SCL、SDA 分别接单片机的 P3.3 口和 P3.1 口(MSP430 单片机只有 P3.3 口和 P3.1 口具有 IIC 总线接口,此处不能接错,更不能任意改接单片机的其他端口)。在具体设计过程中,需要注意的是 MSP430 单片机工作电压为 3.3 V,而 AT24C02 存储器工作电压必须是 5 V。

2.2　矩阵式键盘电路的设计

基于 MSP430 单片机的控制电路中矩阵式键盘电路用于实现设定数据与读写的功能。本设计采用 4 * 4 矩阵式键盘,其中低于 10 的为数值键,大于 10 的为功能键。当设定好数据后,通过功能键与 IIC 总线把数据写入或读取到 AT24C02 存储器中。

2.3　显示电路的设计

常用的显示元器件有数码管、液晶 1602 和液晶 12864,其中液晶 1602 只能显示 ASCI 码[5],而液晶 12864 不仅能显示 ASCI 码,还可以显示汉字。由于后两种显示元器件都存在成本高等缺点,而本文只是探讨 IIC 总线通信,故采用成本低的数码管显示。

3　软件系统设计

软件系统主要由键盘扫描程序、键盘处理程序、显示程序和 IIC 通信程序组成。

(1)IIC 通信字节写入格式

对于具有 IIC 总线的 AT24C02 芯片,其字节写入格式如图 3 所示。

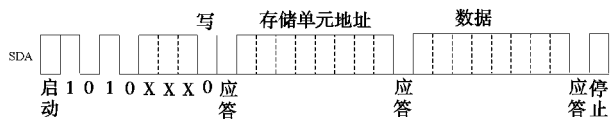


图 3　AT24C02 芯片字节写入格式

Fig. 3　The format of AT24C02 chip byte Write

用函数来描述,用模拟 IIC 时序方法描述如下。

```
void eeprom_write(unsigned char address,unsigned char data)
{
I2C_Start(); //起始信号
I2C_SendByte(0xa0); //写元器件地址
I2CSendACK(); //等待应答
I2C_SendByte(address);//写元器件中存储器地址
I2CSendACK(); //等待应答
```

```
I2C_SendByte(data); //在存储器中写入数据
I2CSendACK(); //等待应答
I2C_Stop(); //停止信号
delayms(2);
}
```

由于 A2A1A0 都接地,3 位引脚编码为 000;由于是写指令,R/W(——)为“0”。故具有 I2C 总线的 AT24C02 芯片 SLA 地址为 10100000,其十六进制就是 0xa0。

用寄存器方法描述如下。

```
void Single_WriteI2C(unsigned char nAddr, unsigned char nVal)
{
I2CNDAT = 0x02; // 发送 2 字节
U0CTL | = MST; // 主机模式
I2CTCTL | = I2CSTT + I2CSTP + I2CTRX; //发送初始化,包括起动信号、停止信号和发送信号
while ((I2CIFG & TXRDYIFG) = = 0); //等待发送准备好
I2CDRB = nAddr; // 装载数
delayms(9);
while ((I2CIFG & TXRDYIFG) = = 0); //等待发送准备好
I2CDRB = nVal; // 装载数
delayms(9);
while ((I2CTCTL & I2CSTP) = = 0x02); // 等待停止信号
}
```

(2)IIC 通信字节读格式

对于具有 I^2C 总线的 AT24C02 芯片,其字节读操作格式如图 4 所示。

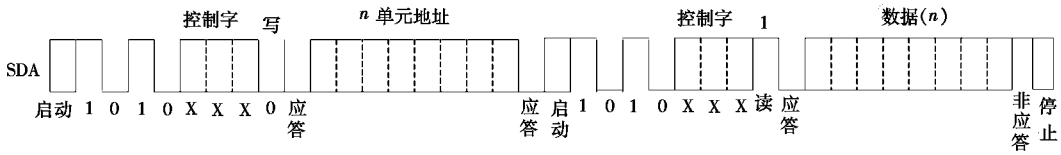


图 4 AT24C02 芯片字节读操作格式

Fig. 4 The format of AT24C02 chip byte Read

用函数来描述,用模拟 IIC 时序方法描述如下。

```
unsigned char eeprom_read(unsigned char address)
{
unsigned char data;
I2C_Start(); // 起动信号
I2C_SendByte(0xa0); //写元器件地址
I2CSendACK(); //等待应答
I2C_SendByte(address);//写元器件中存储器地址
I2CSendACK(); //等待应答
I2C_Start(); //起动信号
I2C_SendByte(0xa1); //读元器件地址
I2CSendACK(); //等待应答
data = I2C_RecvByte(); //读取数据
I2C_Stop(); //停止信号
return data;
}
```

图 4 第 3 个字节的最后一位,由于是“读”操作,所以是“1”,故具有 I2C 总线的 AT24C02 芯片 SLA 地址为 1010 0001,其十六进制就是 0xa1。

用寄存器的方法描述如下。

```
uchar Single_ReadI2C(unsigned char nAddr)
{
I2CNDAT = 0x01; // 发送 1 字节
unsigned char ctlbyte;
U0CTL | = MST; // 主机模式
I2CTCTL | = I2CSTT + I2CSTP + I2CTRX; //发送初始化,包括起动信号、停止信号和发送信号
while ((I2CIFG & TXRDYIFG) = = 0); //等待发送准备好
I2CDRB = nAddr; // 装载数
delayms(9);
U0CTL | = MST; // 主机模式
I2CIFG & = ~ARDYIFG; //清除接收标志位
I2CTCTL & = ~I2CTRX; //接收模式
```

I2CTCTL = I2CSTT + I2CSTP; //起始信号和停止信号

while ((I2CIFG & RXRDYIFG) = = 0); //等待接收准备好

ctlbyte = I2CDRB; // 装载数

delayms(9);

while ((I2CTCTL & I2CSTP) = = 0x02); // 等待停止信号

return ctlbyte;

}

(3)IIC 总线数据传输的起始和停止

启动 IIC 总线,SDA 出现由低至高的转换后将停止数据传输,如图 5 所示。起始和终止信号通常由主机产生,IIC 总线的信号时序有严格的规定。

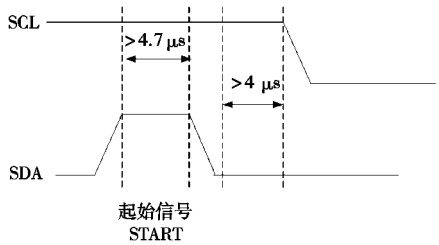


图 5　I2C 总线数据传输的起始信号时序图

Fig. 5　The timing diagram of start to data transmission on I2C bus

对照图 5 可以看出,起始函数可以用模拟 IIC 时序语句描述。

void I2C_Start( )

{

SCLOUT; //SCL 输出

SDAOUT; //SDA 输出

SDA1; //SDA 输出高电平

delayus(5); // 延时 5us

SCL1; //SCL 输出高电平

delayus(5); // 延时 5us

SDA0; //SDA 输出低电平

delayus(5);

}

用寄存器的方法描述如下。

I2CTCTL | = I2CSTT;

IIC 总线数据传输的停止时序图如图所 6 示。

对照图 6 可以看出,停止函数可以用模拟 IIC 时序语句描述。

void I2C_Stop( )

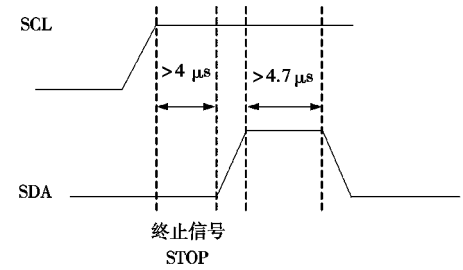


图 6　IIC 总线数据传输的停止信号时序图

Fig. 6　The timing diagram of stop to data transmission on IIC bus

{

SCLOUT; //SCL 输出

SDAOUT; //SDA 输出

SDA0; //SDA 输出低电平

delayus(5); // 延时 5us

SCL1; // SCL 输出高电平

delayus(5); // 延时 5us

SDA1; //SDA 输出高电平

delayus(5);

}

用寄存器的方法描述如下。

I2CTCTL | = I2CSTP;

(4)IIC 通信对总线的“写”操作

对总线的“写”操作,用模拟 IIC 时序方法描述如下。

void I2C_SendByte(uchar dat)

{

uchar i;

SDAOUT; SDA 输出

SCLOUT; SCL 输出

SCL0; SCL 拉低

for (i = 0; i < 8; i + +) //1 字节位传输,先传输高位

{

if((dat&0x80)) //“写”的数据同 0x80 相与,如果是“1”

{

SDA1; //SDA 为高电平

}

else //如果非“1”(为“0”)

{

SDA0; //SDA 为低电平

}

```
dat = dat < <1; //数据左移
delayus(6);
SCL1; //SCL 拉高
delayus(6);
SCL0; //SCL 拉低
delayus(6);
}
SDA1; //SDA 拉高
delayus(3);
}
```

用寄存器的方法描述如下。

```
I2CTCTL | = I2CTRX; //写模式
while ((I2CIFG & TXRDYIFG) = = 0); //等待发送准备好
I2CDRB = nAddr; // 装载数据
```

I2CDRB = nAddr 是把要“写”的“数据”或“地址”nAddr 送入 I2CDRB 寄存器, while ((I2CIFG & TXRDYIFG) = = 0) 表示等待发送准备好。

(5)IIC 通信对总线的“读”操作

对总线的读写都是高位在前,低位在后,一位一位传送的,下面用模拟 IIC 时序程序来描述。

```
uchar I2C_RecvByte( )
{
uchar i,dat;
SDAOUT; //SDA 输出
SCLOUT; //SCL 输出
SCL0; //SCL 拉低
delayus(4); //延时 4us
SDA1; //SDA 拉高
SDAIN; //读 SDA 电平
for (i =0; i <8; i + + ) // 1 字节位读,先读高位
{
SCL1; //SCL 拉高
delayus(2);
dat < < = 1; //左移
if(SDADATA) //如果读 SDA 的电平是“1”,数据加 1;如果读 SDA 的电平是“0”,数据保持不变
{
dat + + ; //数据 +1
}
SCL0; //SCL 拉低
delayus(2);
}
return dat; //返回数据
}
```

用寄存器的方法描述如下。

```
I2CTCTL & = ~I2CTRX; //读模式
I2CTCTL = I2CSTT + I2CSTP;
//起始信号和停止信号
while ((I2CIFG & RXRDYIFG) = = 0); //等待读准备好
ctlbyte = I2CDRB; // 装载数据
```

其实,这个“读”函数关键语句就是 I2CTCTL & = ~I2CTRX; ctlbyte = I2CDRB 表示读取寄存器 I2CDRB 的内容, while ((I2CIFG & RXRDYIFG) = = 0) 表示读已准备好。

(6) 采用中断方法进行 IIC 通信

其中断函数如下。

```
#pragma vector = USART0TX_VECTOR //中断函数
__interrupt void I2C_ISR(void)
{
switch( I2CIV )
{
case 2: break; // 仲裁失败
case 4: break; // 无应答
case 6: break; // 本机地址
case 8: break; // 寄存器访问准备
case 10: // 接收数据
while ((I2CIFG & RXRDYIFG) = = 0);
ctlbyte =I2CDRB;
I2CIFG& = ~RXRDYIFG;
I2CIE& = ~RXRDYIE;
break;
case 12: // 发送数据
I2CTCTL | = I2CSTP;
while ((I2CIFG & TXRDYIFG) = = 0); //等待发送准备好
delayus(3);
I2CDRB =nAddr2; // 装载数据
delayms(9);
while ((I2CIFG & TXRDYIFG) = = 0);
// 等待发送准备好
I2CDRB = nVa2;
I2CTCTL | = I2CSTP;
```

```
while ((I2CTCTL & I2CSTP) == 0x02);
break;
case 14: break; // 广播呼叫
case 16: break; // 起始信号
default:break;
}
I2CIFG& = ~TXRDYIFG; //清除发送中断标志位
I2CIFG& = ~RXRDYIFG;//清除接收中断标志位
}
```

将上述 3 种程序比较,可以看出寄存器方法明显比模拟端口方法方便得多,向存储单元写数据就是通过 I2CTCTL | = I2CSTT + I2CSTP + I2CTRX 语句来实现的,读数据就是通过 I2CTCTL & = ~ I2CTRX, I2CTCTL = I2CSTT + I2CSTP 语句来实现的[6]。在进行读写时候,还必须确定读写的字节数,比如通过 I2CNDAT = 0x01 语句来进行设置。I2CTCTL = I2CSTT + I2CSTP 这条语句就包括起始信号和停止信号语句。至于应答语句, I2CNDAT = 0x02 和 while ((I2CIFG & TXRDYIFG) = =0)语句就包含了应答信号。

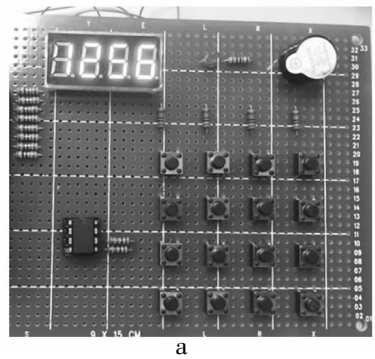
中断方法调试比寄存器方法复杂得多,两者在 IIC 初始化程序就有差别,前者必须设定 SCL 高低电平时间;其次在写函数(void Single _ WriteI2C)里,中断方法特别要注意有参数传递语句(nAddr2 = nAddr;nVa2 = nVal),否则在中断函数里,写函数的参数(nAddr,nVal)就变为 0。最后需要注意的是 nAddr, nVal, ctlbyte, nAddr1, nVa2,nAddr2 均为全局变量。

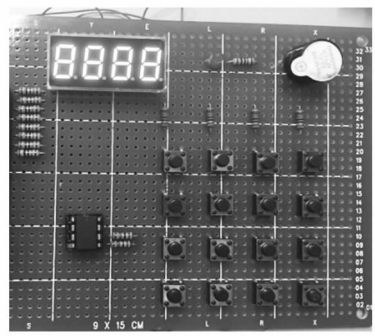
需要指出的是,51 系列单片机由于没有 IIC(I2C)总线端口,因此在使用 IIC(I2C)的时候,必须模拟 IIC 时序,通过 SCL、SDA 端口电平拉高或拉低控制,一位一位地传字节送。而 MSP430F169 单片机的端口具有 IIC 总线,通过对寄存器的设置就能完成字节的传输,且在此过程中,SCL、SDA 端口能够自动调节。

总之,用模拟 IIC 时序方法易懂但繁琐,用寄存器或中断方法方便但稍微难理解些。

4　实验结果

为了验证以上分析结果,根据系统硬件和软件设计,做成实物进行试验,结果如图 7 所示:


a


b

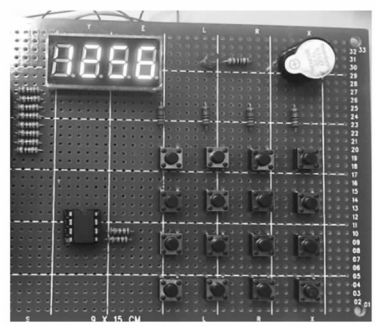

c

图 7　I^2C 总线实验效果图

Fig. 7　The effect picture to I^2C bus experiment

图 7(a)是从 AT24C02 写入的数据 1256;图 7(b)是按下确认键后数码管显示的数据;图 7(c)是从 AT24C02 读出的数据。

5　结论

本文在中高档 M0P430 单片机 IIC 系统通信原理的基础上,充分利用 MSP430 单片机寄存器设置和中断设置等的特点和优良性,通过硬件与软件相结合的办法对存储器电路、键盘电路和显示电路进行了分析和设计,并分别采用模拟 IIC 时序方法、寄存器方法和中断方法进行主机和从机的 IIC 通信,然后对出现的各种问题提出新的解决方法,并通过实验进行验证,表明基于 MSP430 单片机的 IIC 总线通信设计的正确性。

(下转第 63 页)