
USB2ISP

中文数据手册

版本：V2D

USBIO TECH.

USB总线转接芯片：USB2ISP

更新日期：2009.04.30

USB2ISP实现：

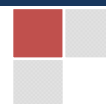
- ✓ USB总线转EPP模式8位并行总线；
- ✓ USB总线转MEM模式8位并行总线；
- ✓ USB总线转标准4线制SPI同步串行总线；
- ✓ USB总线转2线制I2C/IIC/TWI/SMBUS同步串行总线。

更多内容请参考：

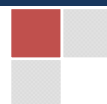
<http://www.usb-i2c-spi.com/cn>

目录

2.1. 特点-概述	5
2.2. 特点-并口	5
2.3. 特点-同步串口	5
3.1. 引脚图.....	6
3.2. 封装形式	6
4.1. 引脚说明-一般说明.....	6
4.2. 引脚说明-公共引脚.....	6
4.3. 引脚说明-并口方式的引脚	7
4.4. 引脚说明-同步串口方式的引脚	8
5.1. 功能说明-一般说明.....	9
5.2. 功能说明-硬件说明.....	9
5.3. 功能说明-厂商 ID 和产品 ID	9
5.4. 功能说明-2 线制 I2C/IIC/TWI/SMBUS 同步串行总线.....	10
5.5. 功能说明-4 线制 SPI 同步串行总线.....	10
5.6. 功能说明-5 线制 SPI 同步串行总线.....	11
5.7. 功能说明-EPP 并行接口.....	11
5.8. 功能说明-MEM 并行接口	12
6.1. 参数-绝对最大值.....	13
6.2. 参数-电气参数	13
6.3. 参数-基本时序参数.....	14
6.4. 参数-MEM 方式时序参数.....	14



6.5 参数-EPP 方式时序参数.....	15
7.1. 应用-基本连接	16
7.2. 应用-2 线制 I2C/IIC/TWI/SMBUS 同步串口应用	17
7.3. 应用-4 线制 SPI 同步串口应用	18
7.4. 应用-5 线制 SPI 同步串口应用	20
7.5. 应用-EPP 并口方式应用.....	21
7.6. 应用-MEM 并口方式应用	22
8.1、关于电容和晶振	24
8.2、关于中断设置的说明.....	24
8.3、SPI 不可以随便改速率.....	24
8.4、I2C 接口上拉电阻	25
8.5、USB2ISP 外围元器件说明	25
8.6、数据缓冲区是否必须限制 4096.....	25
8.7、不能识别 USB 检测.....	25
9.1、Windows 系统下的驱动	26
9.2、Linux 系统下的驱动.....	30
10.1. 设备管理 API.....	31
10.2. 中断处理 API.....	32
10.3. 并口数据传输 API.....	33
10.4. 同步串口数据传输 API	35



1、概述

USB2ISP是一个USB总线的转接芯片。USB2ISP实现：

- ✓ USB总线转EPP模式8位并行总线；
- ✓ USB总线转MEM模式8位并行总线；
- ✓ USB总线转标准4线制SPI同步串行总线，包括SCK，MISO，MOSI，CS0，CS1，CS2信号线；
- ✓ USB总线转2线制I2C/IIC/TWI/SMBUS同步串行总线，包括SCL和SDA信号线。

USB2ISP是USB2I2C和USB2SPI的合集，如果在设计中只需要USB转I2C功能请直接使用USB2I2C（SSOP20封装）；如果只使用USB转SPI功能请直接使用USB2SPI（SSOP20封装）。

在并口模式下，USB2ISP提供了EPP方式或MEM方式的8位主动并行接口，实现PC上位机和下位控制器之间的直接数据输入输出，而不再需要单片机/DSP/MCU等的监控。

在同步串口方式下，USB2ISP芯片支持4线制SPI同步串口，提供3条CS线（CS0，CS1，CS2）、SCK线、DIN（MISO）线和DOUT（MOSI）线，非常方便地实现对SPI类接口的设备读写；USB2ISP芯片还支持2线制I2C/IIC/TWI/SMBUS同步串口，提供SCL线和SDA线，PC上位机可以方便地对I2C/IIC/TWI/SMBUS接口器件进行读写。

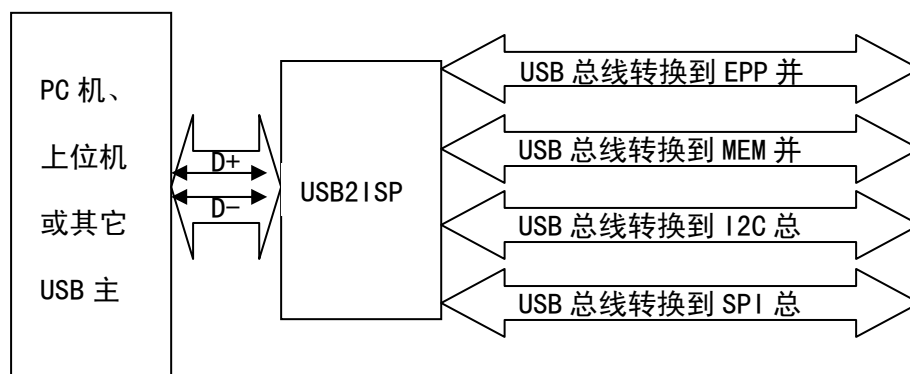


图 1- USB2ISP 功能结构

2、特点

2.1. 特点-概述

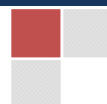
- ✓ 全速USB设备接口, 兼容USB V2.0, 外围元器件只需要1个12M晶体和2个电容。
- ✓ 低成本, 直接转换原并口外围设备、原SPI接口外围设备、I2C接口外围设备。
- ✓ 采用SOP-28封装, 2线制I2C/IIC/TWI/SMBUS同步串口应用还提供小型的SSOP-20封装。
- ✓ 由于是通过USB转换的接口, 所以只能做到应用层兼容, 而无法绝对相同。

2.2. 特点-并口

- ✓ 提供两种接口方式: EPP方式和MEM方式。
- ✓ EPP方式提供AS#、DS#、WR#等信号, 类似于EPPV1.7或EPPV1.9。
- ✓ MEM方式提供A0、RD#、WR#等信号, 类似于存储器读写方式。

2.3. 特点-同步串口

- ✓ 采用FlexWire™技术, 通过软件能够实现灵活多样的2线到5线的同步串口。
- ✓ 作为Host/Master主机端, 支持2线和4线等常用的同步串行接口。
- ✓ 2线制I2C/IIC/TWI/SMBUS接口, 支持4种传输速度: 20KHz/100KHz/400KHz/750KHz。
- ✓ 4线制SPI同步串行接口, 750KHz固定时钟频率。



3、引脚图、封装

3.1. 引脚图

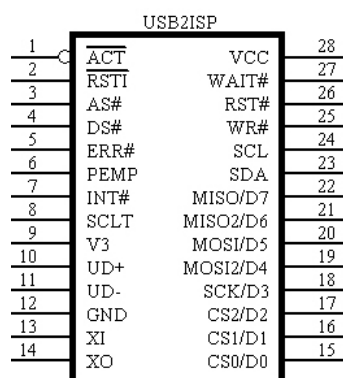


图 2 - USB2ISP 引脚图

3.2. 封装形式

封装形式	塑体宽度		引脚间距		芯片厚度		封装说明	订货型号
SOP-28	7.62mm	300mil	1.27mm	50mil	2.54mm	100mil	标准的28脚贴片	USB2ISP

4、引脚说明

4.1. 引脚说明-一般说明

USB2ISP的具体功能由复位后的功能配置决定, 同一引脚在不同功能下的定义可能不同。

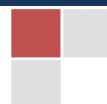
4.2. 引脚说明-公共引脚

引脚号	引脚名称	类型	引脚说明
-----	------	----	------

28	VCC	电源	正电源输入端, 需要外接0.1 μ F电源退耦电容
12	GND	电源	公共接地端, 直接连到USB总线的地线
9	V3	电源	在3.3V电源电压时连接VCC输入外部电源, 在5V电源电压时外接容量为0.01 μ F退耦电容
13	XI	输入	晶体振荡的输入端, 需要外接晶体及振荡电容
14	XO	输出	晶体振荡的反相输出端, 需要外接晶体及振荡电容
10	UD+	双向三态	直接连到USB总线的D+数据线, 内置上拉电阻
11	UD-	双向三态	直接连到USB总线的D-数据线
1	ACT#	输出	USB设备配置, 通过2K欧电阻下拉到地
2	RST I	输入	外部复位输入, 高电平有效, 内置下拉电阻

4.3. 引脚说明-并口方式的引脚

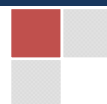
引脚号	引脚名称	类型	引脚说明
22~15	D7~D0	双向三态	8位双向数据总线, 内置上拉电阻
25	WR#	输出	EPP方式: 写操作指示, 低电平写, 高电平读
			MEM方式: 写选通输出WR#, 低电平有效
4	DS#	输出	EPP方式: 数据操作选通, 低电平有效
			MEM方式: 读选通输出RD#, 低电平有效
26	RST#	输出	复位输出, 低电平有效
3	AS#	三态输出	EPP方式: 地址操作选通, 低电平有效
			MEM方式: 地址线输出
27	WAIT#	输入	请求等待, 低有效, 内置上拉
7	INT#	输入	中断请求输入, 上升沿有效, 内置上拉电阻
5	ERR#	输入	自定义通用输入, 内置上拉电阻



8	SLCT	输入	自定义通用输入, 内置上拉电阻
6	PEMP	输入	自定义通用输入, 内置上拉电阻

4.4. 引脚说明-同步串口方式的引脚

引脚号	引脚名称	类型	引脚说明
22	DIN (MISO)	输入	4线串口数据输入, 内置上拉电阻
21	DIN2 (MISO2)	输入	5线串口数据输入2, 内置上拉电阻
20	DOUT (MOSI)	三态输出	4线串口数据输出
19	DOUT2 (MOSI2)	三态输出	5线串口数据输出2
18	DCK	三态输出	4线/5线串口时钟输出
17	CS2	三态输出	4线串口片选输出2#
16	CS1	三态输出	4线串口片选输出1#
15	CS0	三态输出	4线串口片选输出0#
24	SCL	开漏输出	2线串口的时钟输出, 内置上拉电阻
23	SDA	开漏输出 及输入	2线串口的数据输入输出, 内置上拉电阻
26	RST#	输出	复位输出, 低电平有效
7	INT#	输入	中断请求输入, 上升沿有效, 内置上拉电阻



5、功能说明

5.1. 功能说明-一般说明

本手册中的数据, 后缀B为二进制数, 后缀H为十六进制数, 否则为十进制数。

USB2ISP的并口是主动式并口, 在计算机端的程序控制下, 可以直接从外部电路输入输出数据, 一般不需要外接单片机/DSP/MCU。

USB2ISP的并口有2种接口方式: EPP方式和MEM方式。EPP方式类似于EPPV1.7或者EPPV1.9, MEM方式类似于INTEL时序存储器的读写方式。芯片复位后的默认方式是EPP, 在USB配置完成后, 计算机端的程序可以随时控制USB2ISP在上述2种方式之间进行切换。

5.2. 功能说明-硬件说明

USB2ISP芯片内置了USB上拉电阻, UD+和UD-引脚应该直接连接到USB总线上。

USB2ISP芯片正常工作时需要外部向XI引脚提供12MHz的时钟信号。一般情况下, 时钟信号由USB2ISP内置的反相器通过晶体稳频振荡产生。外围电路只需要在XI和X0引脚之间连接一个12MHz的晶体, 并且分别为XI和X0引脚对地连接振荡电容。

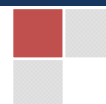
USB2ISP芯片内置了电源上电复位电路。RSTI引脚用于从外部输入异步复位信号: 当RSTI引脚为高电平时, USB2ISP芯片被复位; 当RSTI引脚恢复为低电平后, USB2ISP会继续延时复位20ms左右, 然后进入正常工作状态。为了在电源上电期间可靠复位并且减少外部干扰, 建议在RSTI引脚与VCC之间跨接一个容量为0.47uF左右的电容。RST#引脚是复位输出引脚, 当其为低电平时, 说明USB2ISP芯片正在复位或者计算机端的程序要求复位外部电路。INT#引脚是中断请求输入引脚, 当其检测到上升沿时, 计算机端的程序将会收到中断通知。其它引脚都是自定义的通用输入引脚, 计算机端的应用程序可以查询其引脚状态。

USB2ISP芯片的部分引脚具有多个功能, 所以在芯片复位期间与复位完成后的正常工作状态下具有不同的特性。所有类型为三态输出的引脚, 都内置了上拉电阻, 在芯片复位完成后作为输出引脚, 而在芯片复位期间三态输出被禁止, 由内置的上拉电阻提供上拉电流。如果必要, 外部电路可以在电路中再提供外置的上拉电阻或者下拉电阻, 从而设定相关引脚在USB2ISP芯片复位期间的默认电平, 外置上拉电阻或者下拉电阻的阻值通常在 $2K\Omega \sim 5K\Omega$ 之间。例如, 并口方式下的AS#在芯片复位期间三态输出禁止, 仅靠上拉电流维持高电平, 为了避免外部电路在此期间受到干扰而误动作, 可以加阻值为 $3K\Omega$ 的上拉电阻, 以维持较稳定的高电平。

USB2ISP芯片使用5V电源电压时, V3引脚应该外接容量为0.01uF左右的电源退耦电容。

5.3. 功能说明-厂商ID和产品ID

USB2ISP默认的厂商ID和产品ID为5512H。



5.4. 功能说明-2 线制 I2C/IIC/TWI/SMBUS 同步串行总线

由USB2ISP转换的2线制同步串口I2C/IIC/TWI/SMBUS是主动式串口, 只能作为I2C/IIC/TWI/SMBUS总线上的Host或Master主机端, 在计算机端的程序控制下, 可以直接从外部电路输入输出数据, 一般不需要外接单片机/DSP/MCU。

2线制同步串口I2C/IIC/TWI/SMBUS的主要引脚包括SCL引脚、SDA引脚。SCL用于单向输出同步时钟, 开漏输出且内置上拉电阻, SDA用于准双向数据输入输出, 开漏输出及输入且内置上拉电阻。

2线制同步串口I2C/IIC/TWI/SMBUS的基本操作元素包括: 起始位、停止位、位输出、位输入。

起始位定义为当SDA为高电平时, SCL输出下降沿(从高电平切换为低电平)。

停止位定义为当SDA为高电平时, SCL输入上升沿(从低电平切换为高电平)。

位输出定义为当SCL为低电平时, SDA输出位数据, 然后SCL输出高电平脉冲。

位输入定义为SCL输出高电平脉冲, 在下降沿之前从SDA输入位数据。

字节输出定义为8个位输出及1个位输入用于应答。

字节输入定义为8个位输入及1个位输出用于应答。

I2C总线的数据输入和输出以字节为单位, 每个字节含8个位, 高位在前。

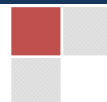
USB2ISP的2线制同步串口支持大部分的标准或兼容I2C/IIC/TWI/SMBUS的设备, 如:

- ✓ I2C接口EEPROM: 24C01A到24C16、24C32到24C1024等;
- ✓ I2C总线8位并行IO口扩展芯片PCF8574/JLC1562;
- ✓ I2C接口实时时钟芯片DS1307/PCF8563/SD2000D/M41T80/ME901/ISL1208/;
- ✓ I2C数据采集ADC芯片MCP3221(12bitADC)/ADS1100(16bitADC)/ADS1112(16bitADC)/MAX1238(12bitADC)/MAX1239(12bitADC);
- ✓ I2C接口数模转换DAC芯片DAC5574(8bitDAC)/DAC6573(10bitDAC)/DAC8571(16bitDAC);
- ✓ I2C接口温度传感器TMP101/TMP275/DS1621/MAX6625, 等

5.5. 功能说明-4 线制 SPI 同步串行总线

由USB2ISP转换的4线制SPI同步串口是主动式串口, 只能作为Host/Master主机端, 在计算机端的程序控制下, 可以直接从外部电路输入输出数据, 一般不需要外接单片机/DSP/MCU。

4线SPI同步串口的主要引脚包括DCK引脚、DIN(MISO)引脚、DOUT(MOSI)引脚、片选引脚CS0、CS1、CS2。DCK用于单向输出同步时钟, DIN(MISO)用于单向输入数据, DOUT(MOSI)用于单向输出数据,



片选引脚CSn (CS0, CS1, CS2) 用于选择设备。

4线串口的基本操作元素包括：片选选中、片选结束、位输出、位输入。片选选中定义为片选引脚CSn输出有效电平（可以定义为高电平或低电平）。片选结束定义为片选引脚CSn输出非有效电平。位输出定义为当DCK为低电平时，DOUT (MOSI) 输出位数据，然后DCK输出高电平脉冲。位输入定义为DCK输出高电平脉冲，在下降沿之后从DIN (MISO) 输入位数据。字节输出定义为8个位输出，字节输入定义为8个位输入。4线串口的数据输入和输出以字节为单位，每个字节含8个位，支持低位在前和高位在前。

USB2ISP目前的版本只支持4线制SPI串行总线的Model0模式。

通过USB2SPI芯片可以非常方便地实现PC机USB总线和下位机端各种SPI接口设备之间的通信，例如：

- ✓ SPI接口EEPROM存储器，ATMEL公司的AT93CXX系列；
- ✓ SPI接口ADC芯片TLC0831 (8bitADC) /TLC1549 (10bitADC) /ADS7816 (12bitADC) /TLV2543 (12bitADC) /TLC2543 (12bitADC) ；
- ✓ SPI接口数模转换DAC芯片TLC7528 (8bitDAC) /TLC5615 (10bitDAC) /TLC5613 (16bitDAC) /；
- ✓ SPI接口温度传感器TMP125/TC77/ADT7301，等。

5.6. 功能说明-5 线制 SPI 同步串行总线

由USB2ISP转换的5线制SPI同步串口是主动式串口，只能作为Host/Master主机端，在计算机端的程序控制下，可以直接从外部电路输入输出数据，一般不需要外接单片机/DSP/MCU。

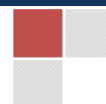
5线SPI同步串口的主要引脚包括DCK引脚、DIN (MISO) 引脚、DIN (MISO) 2引脚、DOUT (MOSI) 引脚、DOUT (MOSI) 2引脚、片选引脚CS0、CS1、CS2。DCK用于单向输出同步时钟，DIN (MISO) 和DIN (MISO) 2用于单向输入数据，DOUT (MOSI) 和DOUT (MOSI) 2用于单向输出数据，片选引脚CSn用于选择外部设备。

5线串口是4线串口中增加一组数据输入DIN (MISO) 2和一组数据输出DOUT (MOSI) 2的版本。5线串口的基本操作元素包括：片选选中、片选结束、双位输出、双位输入。双位输出定义为当DCK为低电平时，DOUT (MOSI) 和DOUT (MOSI) 2输出位数据，然后DCK输出高电平脉冲。一个字节8位数据，高4位从DOUT (MOSI) 输出，低4位从DOUT (MOSI) 2输出。双位输入定义为DCK输出高电平脉冲，在下降沿之后从DIN (MISO) 和DIN (MISO) 2输入位数据。一个字节8个数据，高4位来自DIN (MISO) 输入，低4位来自DIN (MISO) 2输入。字节输出定义为一个字节8位分为两组各4位分别输出。字节输入定义为分别输入两组各4位组合成一个字节。5线串口的数据输入和输出以半字节为单位，每个半字节含4个位，支持低位在前和高位在前。要分别在DOUT (MOSI) 和DOUT (MOSI) 2产生字节输出，那么需要两个字节输出组合。USB2ISP的5线串口用于仿真具有较多I/O的同步串行接口。

USB2ISP目前的版本只支持5线制SPI串行总线的Model0模式。

5.7. 功能说明-EPP 并行接口

USB2ISP的并口有2种接口方式：EPP方式和MEM方式。EPP方式类似于EPPV1.7或者EPPV1.9，MEM方式类似于INTEL时序存储器的读写方式。芯片复位后的默认方式是EPP，在USB配置完成后，计算机端的程序



可以随时控制USB2ISP在上述2种方式之间进行切换。

EPP并口的主要引脚包括WR#引脚、DS#引脚、AS#引脚、WAIT#引脚,相关信号的时序说明可以参照EPP规范V1.7和V1.9。

EPP方式通过WR#、DS#和AS#的逻辑组合执行具体操作。WR#用于指示当前的数据或地址传输方向,对计算机端而言,高电平是对外部电路执行读操作,低电平是对外部电路执行写操作。选通信号是低电平有效的脉冲信号,选通信号包括数据选通DS#和地址选通AS#,DS#有效执行数据操作,AS#有效执行地址操作。EPP的实际操作发生于选通信号有效期间,例如:在WR#为高电平期间DS#输出脉冲,则执行一个数据读操作;在WR#为低电平期间AS#输出脉冲,则执行一个地址写操作。

USB2ISP支持WAIT#等待信号,在USB2ISP开始输出低电平选通信号后,如果WAIT#为低电平,那么选通信号将继续保持低电平直到WAIT#恢复为高电平才结束输出。

写操作的选通信号的低电平有效宽度最小是0.33 μ S或者0.5 μ S,读操作的选通信号的低电平有效宽度最小是0.5 μ S或者0.66 μ S,理想状态下的最大传输速度是660KB/S,在WINDOWS XP SP2环境下使用连续的大数据块进行速度测试,实测传输速度约为:下传370KB/S,上传560KB/S。

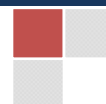
5.8. 功能说明-MEM 并行接口

USB2ISP的并口有2种接口方式:EPP方式和MEM方式。EPP方式类似于EPPV1.7或者EPPV1.9,MEM方式类似于INTEL时序存储器的读写方式。芯片复位后的默认方式是EPP,在USB配置完成后,计算机端的程序可以随时控制USB2ISP在上述2种方式之间进行切换。

MEM并口的主要引脚包括WR#引脚、RD#引脚(DS#引脚的别名)、A0引脚(AS#引脚的别名)、WAIT#引脚。

MEM方式类似于存储器的读写方式,WR#和RD#都是低电平有效的脉冲信号。MEM的实际操作发生于WR#或者RD#有效期间,对计算机端而言,当WR#有效时对外部电路执行写操作,当RD#有效时对外部电路执行读操作。A0用于指示当前读写操作的地址,例如:将A0和A0的反相分别用于两个外部设备的片选;或者将A0=1时的操作指向外部设备的命令端口,而将A0=0时的操作指向数据端口。

WR#的低电平有效宽度最小是0.5 μ S,RD#的低电平有效宽度最小是0.83 μ S,理想状态下的最大传输速度是600KB/S。实测传输速度略低于EPP并口方式。



6、参数

6.1. 参数-绝对最大值

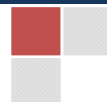
（临界或者超过绝对最大值将可能导致芯片工作不正常甚至损坏）

名称	参数说明	最小值	最大值	单位
TA	工作时的环境温度	-20	70	°C
TS	储存时的环境温度	-55	125	°C
VCC	电源电压（VCC接电源, GND接地）	-0.5	6.5	V
VIO	输入或者输出引脚上的电压	-0.5	VCC+0.5	V

6.2. 参数-电气参数

（测试条件：TA=25°C, VCC=5V, 不包括连接USB总线的引脚）

名称	参数说明	最小值	典型值	最大值	单位
VCC	电源电压（V3引脚不连VCC引脚）	4.5	5	5.3	V
ICC	工作时总电源电流		15	30	mA
ISLP	USB挂起时的总电源电流		0.5		mA
VIL	低电平输入电压	-0.5		0.7	V
VIH	高电平输入电压	2.0		VCC+0.5	V
VOL	低电平输出电压（4mA吸入电流）			0.5	V
VOH	高电平输出电压（4mA输出电流）（芯片复位期间仅100uA输出电流）	VCC-0.5			V
IUPs	SCL和SDA引脚的高电平输出电流	100	200	500	uA
IUP	内置上拉电阻的输入端的输入电流	40	80	160	uA



IDN	内置下拉电阻的输入端的输入电流		-50		uA
VR	电源上电复位的电压门限	2.3	2.6	2.9	V

6.3. 参数-基本时序参数

(测试条件: TA=25℃, VCC=5V)

名称	参数说明	最小值	典型值	最大值	单位
FSCK	XI 引脚的输入时钟信号的频率	11.98	12.00	12.02	MHz
TPR	电源上电的复位时间		20	40	mS
TRI	外部复位输入的有效信号宽度	100			nS
TRD	外部复位输入后的复位延时		30		mS

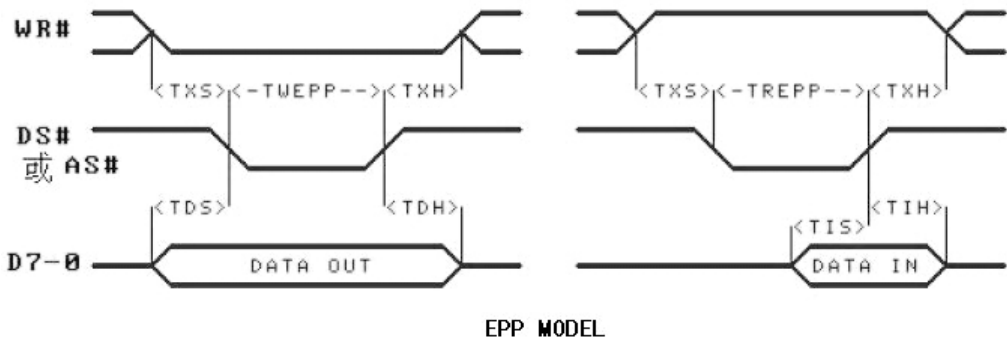


图 3 - EPP 模式时序图

6.4. 参数-MEM 方式时序参数

(测试条件: TA=25℃, 参考附图前半部分)

名称	参数说明	最小值	典型值	最大值	单位
TWMEM	写选通WR#的低电平有效宽度	300	500		nS
TRMEM	读选通RD#的低电平有效宽度	450	830		nS
TAS	WR#或RD#有效前的地址建立时间	450			nS
TAH	WR#或RD#有效后的地址保持时间	300			nS

TDS	WR#有效前的数据输出建立时间	150			nS
TDH	WR#有效后的数据输出保持时间	300			nS
TIS	RD#无效前的数据输入建立时间	350			nS
TIH	RD#无效后的数据输入保持时间	0			nS

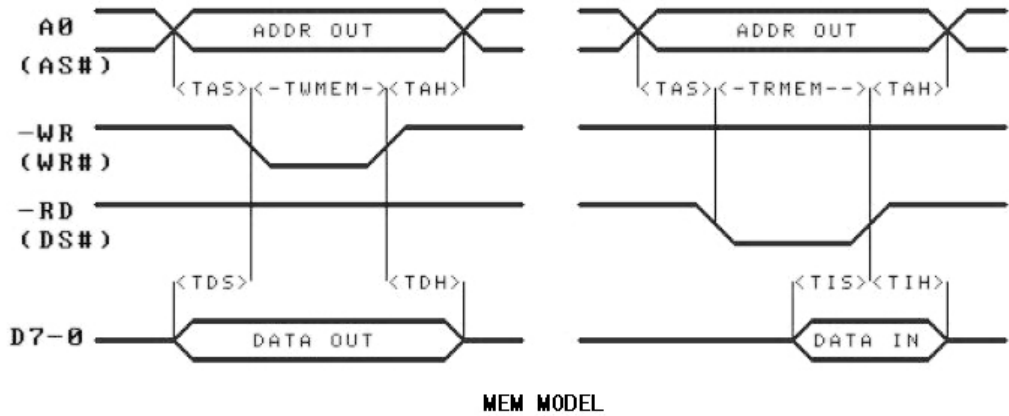


图 4 - MEM 模式时序图

6.5 参数-EPP 方式时序参数

(测试条件: TA=25℃, 参考附图后半部分)

名称	参数说明	最小值	典型值	最大值	单位
TWEPP	写操作的选通的低电平有效宽度	300	500		nS
TREPP	读操作的选通的低电平有效宽度	450	660		nS
TXS	选通有效前的方向WR#建立时间	800			nS
TXH	选通有效后的方向WR#保持时间	300			nS
TDS	选通有效前的数据输出建立时间	150			nS
TDH	选通有效后的数据输出保持时间	450			nS
TIS	选通无效前的数据输入建立时间	350			nS
TIH	选通无效后的数据输入保持时间	0			nS

7、应用

7.1. 应用-基本连接

USB2ISP的基本连接图如下图所示。P3是USB端口, USB总线包括一对5V电源线 and 一对数据信号线, 通常, +5V电源线是红色, 接地线是黑色, D+信号线是绿色, D-信号线是白色。USB总线提供的电源电流最大可以达到500mA, 一般情况下, USB2ISP芯片和低功耗的USB产品可以直接使用USB总线提供的5V电源。如果USB产品通过其它供电方式提供常备电源, 那么USB2ISP也应该使用该常备电源, 如果需要同时使用USB总线的电源, 那么可以通过阻值约为1-3 Ω 的电阻连接USB总线的5V电源线与USB产品的5V常备电源, 并且两者的接地线直接相连接。

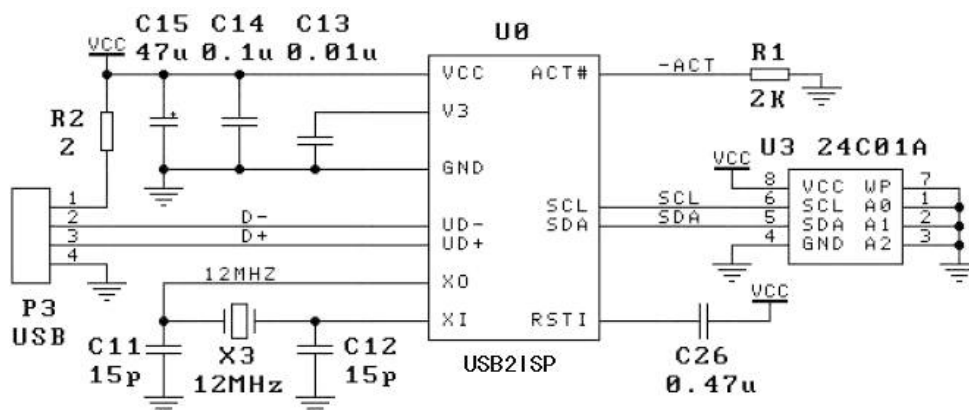


图 5 - USB2ISP 基本连接图

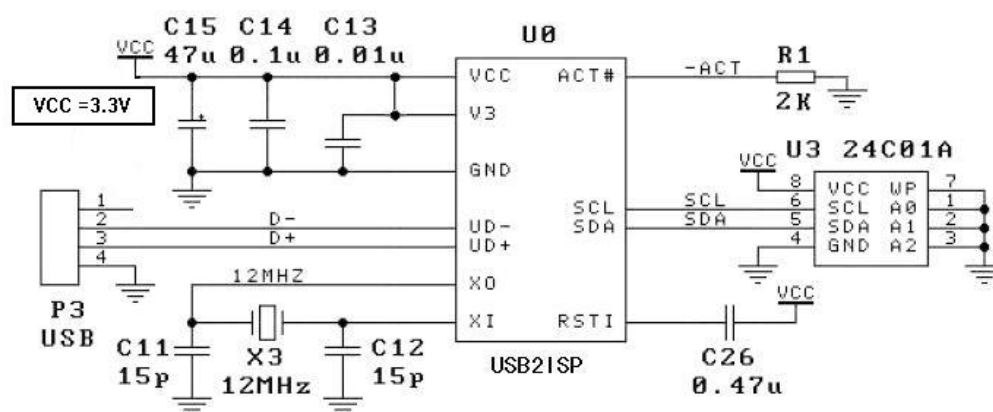
C13和C14是独石或高频瓷片电容, C13容量为1000pF到0.01 μ F, 用于USB2ISP内部电源节点退耦, C14容量为0.1 μ F, 用于外部电源退耦。晶体X3、电容C11和C12用于时钟振荡电路。X3的频率是12MHz, C11和C12是容量为15pF的独石或高频瓷片电容。

如果USB产品使用USB总线的电源, 并且在VCC与GND之间并联了较大的电容C15, 使得电源上电过程较慢并且电源断电后不能及时放电, 那么USB2ISP将不能可靠复位。建议在RSTI引脚与VCC之间跨接一个容量为0.47 μ F的电容C26延长复位时间。R1是配置电阻, ACT引脚必须通过2K欧电阻下拉到地。

在设计印刷线路板PCB时, 需要注意: 退耦电容C13和C14尽量靠近USB2ISP的相连引脚; 使D+和D-信号线贴近平行布线, 尽量在两侧提供地线或者覆铜, 减少来自外界的信号干扰; 尽量缩短XI和X0引脚相关信号线的长度, 为了减少高频干扰, 可以在相关元器件周边环境地线或者覆铜。

外部24XX系列串行EEPROM配置芯片U3是可选器件, 可以方便验证I2C总线操作; 当U3被省去时, 同样可以正常工作, 这是提供SCL和SDA引脚和I2C Device连接。

USB2ISP也可以工作在3.3V系统中, 典型的连接方式如下图所示。和5V系统不同的放在于: V3管脚要和VCC一起链接到3.3V电源上。

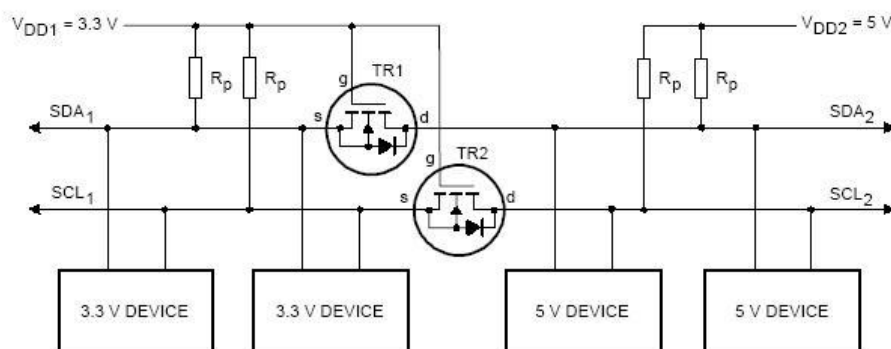


7.2. 应用-2 线制 I2C/IIC/TWI/SMBUS 同步串口应用

2线制I2C/IIC/TWI/SMBUS同步串口应用连接图如下图所示。I2C总线支持多个设备的地址识别,采用数据流方式读写数据,支持一次读写较大的数据块。USB2ISP的I2C两线串口支持20KHz、100KHz、400KHz、750KHz的速度,与具有硬件两线串口的设备连接时可以选择较高的速度,与软件模拟两线串口的单片机连接时只能选择较低的速度(例如20KHz)。

USB2ISP芯片只能作为I2C通信的主机端(Master),而且USB2ISP只能作为I2C总线中的唯一的一个主机端(Master);也就是说USB2ISP只能工作在单主通信的I2C总线里,而且这个主设备就是USB2ISP。

5V系统和3.3V系统是两种常见的I2C通信系统,两种不同的电压系统进行通信时,必须添加电平转换芯片或者采用MOSFET驱动:



I²C 总线系统中连接两个电压不同的部分的双向电平转换器电路

SPI操作的其它配置请参考USB2ISP驱动文件夹【USB2ISP_DRIVER\LIB_C】下面的USBIOX.H文件里面的相关说明(USBIO_SetStream)。

```
USBIO_SetStream( // 设置串口流模式
    ULONG iIndex, // 指定USB2ISP设备序号
    ULONG iMode ); // 指定模式,见下行
```

// 位1-位0: I2C接口速度/SCL频率, 00=低速/20KHz, 01=标准/100KHz(默认值), 10=快速/400KHz, 11=高速/750KHz

// 位2: SPI的I/O数/I/O引脚, 0=单入单出(SCK时钟/MOSI出/MISO入)(默认值),

1=双入双出(SCK时钟/MOSI出MSOSI2出/MISO入MISO2入)

// 位7: SPI字节中的位顺序, 0=低位在前, 1=高位在前

// 其它保留, 必须为0

其它I2C操作还包括:

- ✓ USBIO_ReadI2C;
- ✓ USBIO_WriteI2C;
- ✓ USBIO_ReadEEPROM;
- ✓ USBIO_WriteEEPROM;
- ✓ USBIO_StreamI2C

更详细的说明请参考: http://www.usb-i2c-spi.com/cn/rar/USB2XXX_Mamu.pdf。

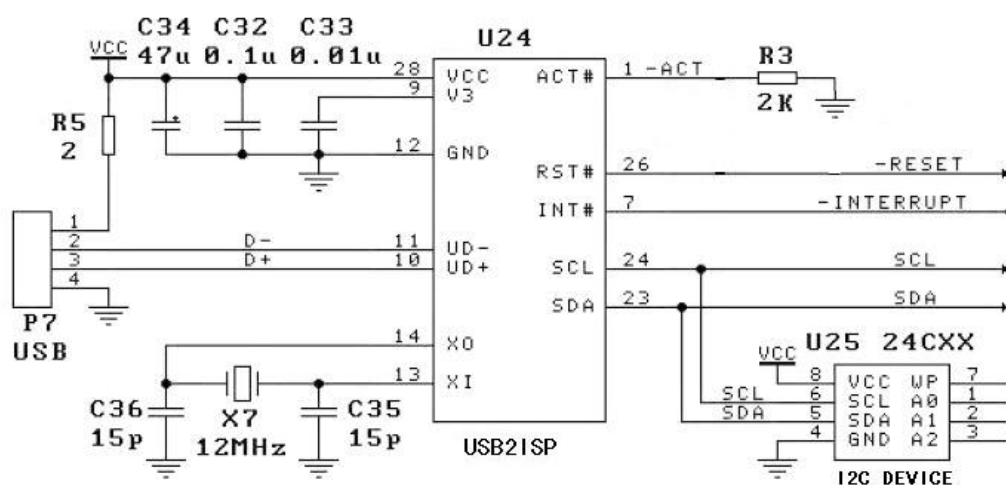


图 6 - USB2ISP 转换成 I2C 接口

7.3. 应用-4 线制 SPI 同步串口应用

4线制SPI同步串口应用连接图如下图所示。标准的4线制SPI同步串口提供SCK时钟线、MISO、MOSI和3条片选线(CS0, CS1, CS2)。SCK时钟频率固定在750KHz, 芯片默认4线制SPI通信工作在模式0(SPI Model 0)。USB2ISP必须作为SPI通信主设备(Master), 与USB2ISP相连的器件必须工作在模式下(Slave)。

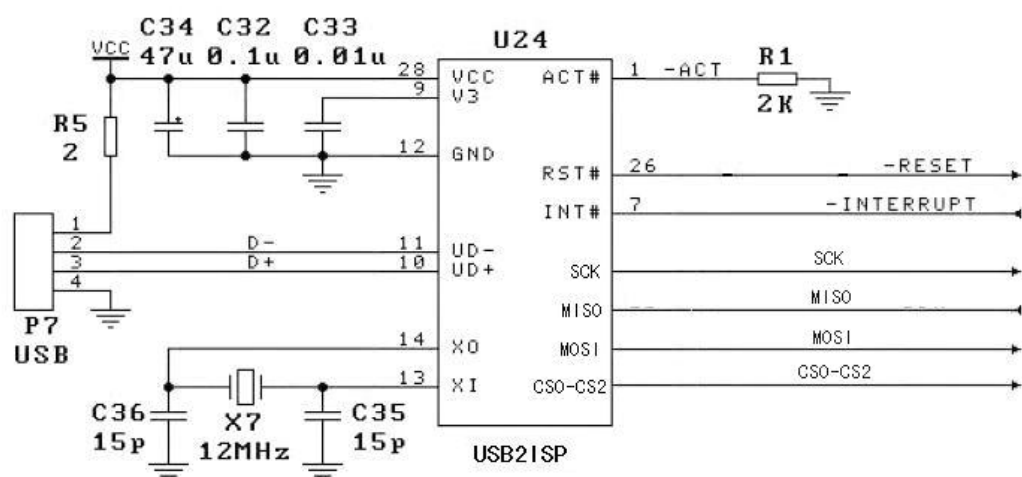
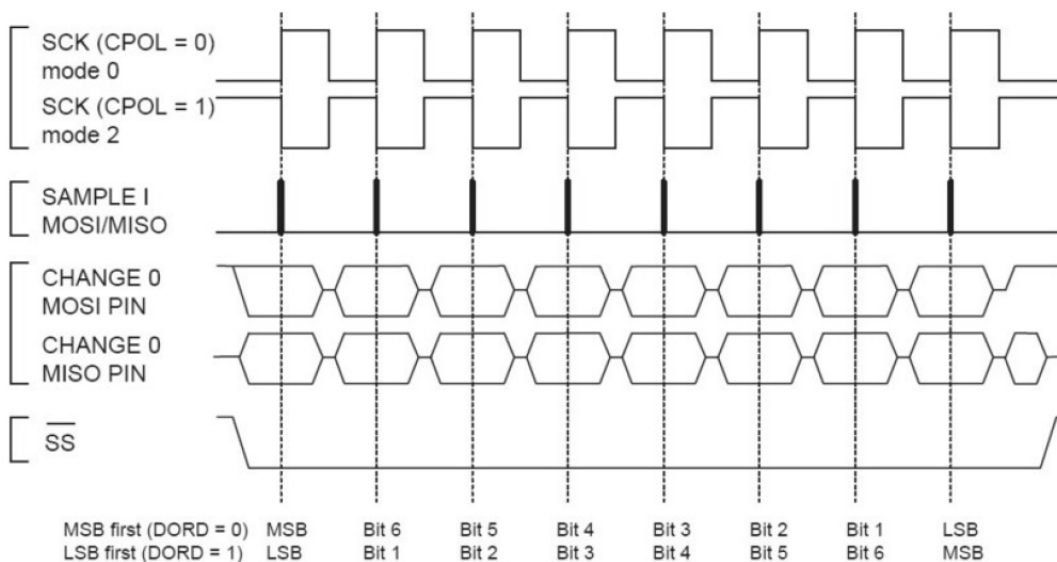


图 7 - USB2ISP 转换成 4 线制 SPI 接口

SPI工作模式参见下表:

SPI Mode	CPOL	CPHA	Shift SCK edge	Capture SCK edge
0	0	0	Falling	Rising
1	0	1	Rising	Falling
2	1	0	Rising	Falling
3	1	1	Falling	Rising

SPI工作模式0时序图如下图所示:



SPI操作的其它配置请参考USB2ISP驱动文件夹【USB2ISP_DRIVER\LIB_C】下面的USBIOX.H文件里面的相关说明(USBIO_SetStream)。4线制SPI通过调用USBIO_StreamSPI4实现。

```
USBIO_SetStream( // 设置串口流模式
```

```
    ULONG    iIndex, // 指定USB2ISP设备序号
```

```

ULONG   iMode ); // 指定模式, 见下行

// 位1-位0: I2C接口速度/SCL频率, 00=低速/20KHz, 01=标准/100KHz (默认值), 10=快速/400KHz, 11=高速/750KHz

// 位2:     SPI的I/O数/I/O引脚, 0=单入单出(SCK时钟/MOSI出/MISO入) (默认值),

                1=双入双出(SCK时钟/MOSI出/MOSI2出/MISO入/MISO2入)

// 位7:     SPI字节中的位顺序, 0=低位在前, 1=高位在前

// 其它保留, 必须为0

更详细的说明请参考: http://www.usb-i2c-spi.com/cn/rar/USB2XXX\_Mamu.pdf.

```

7.4. 应用-5 线制 SPI 同步串口应用

5线制SPI同步串口应用连接图如下图所示。标准的5线制SPI同步串口提供SCK时钟线、MISO、MISO2、MOSI、MOSI2和3条片选线(CS0, CS1, CS2)。SCK时钟频率固定在750KHz, 芯片默认5线制SPI通信工作在模式0 (SPI Mode1 0)。USB2ISP必须作为SPI通信主设备(Master), 与USB2ISP相连的器件必须工作从模式下(Slave)。

5线制SPI通过调用USBIO_StreamSPI5实现。

更详细的说明请参考: http://www.usb-i2c-spi.com/cn/rar/USB2XXX_Mamu.pdf。

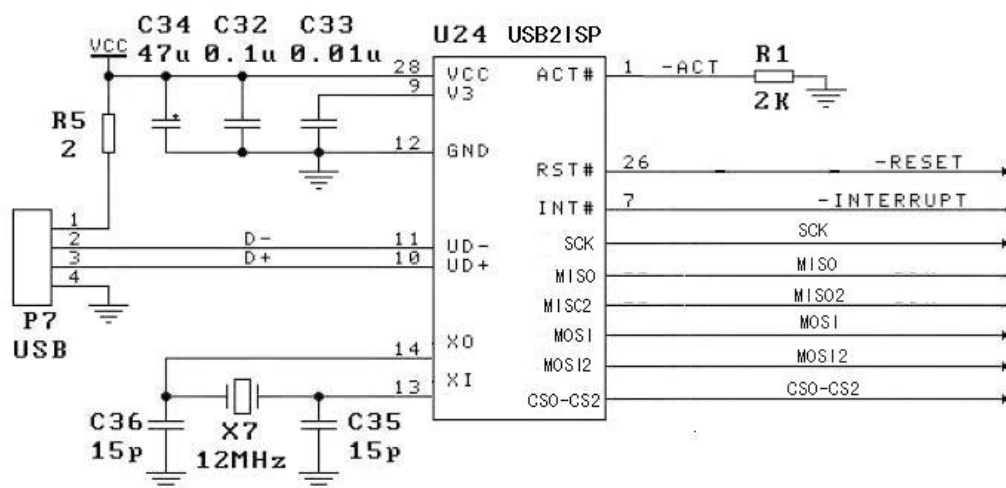


图 8 - USB2ISP 转换成 5 线制 SPI 接口

7.5. 应用-EPP 并口方式应用

EPP 并口的主要引脚包括WR#引脚、DS#引脚、AS#引脚、WAIT#引脚，相关信号的时序说明可以参照EPP 规范V1.7 和V1.9。

EPP 方式通过WR#、DS#和AS#的逻辑组合执行具体操作。WR#用于指示当前的数据或地址传输方向，对计算机端而言，高电平是对外部电路执行读操作，低电平是对外部电路执行写操作。选通信号是低电平有效的脉冲信号，选通信号包括数据选通DS#和地址选通AS#，DS#有效执行数据操作，AS#有效执行地址操作。EPP 的实际操作发生于选通信号有效期间，例如：在WR#为高电平期间DS#输出脉冲，则执行一个数据读操作；在WR#为低电平期间AS#输出脉冲，则执行一个地址写操作。

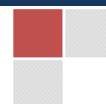
USB2ISP的EPP数据读写操作DS#支持WAIT#等待信号，在USB2ISP开始输出低电平选通信号后，如果WAIT#为低电平，那么选通信号将继续保持低电平直到WAIT#恢复为高电平或者85uS等待超时后才结束输出。

USB2ISP的EPP地址读写操作AS#不支持WAIT#等待信号，所以EPP地址读写操作比数据读写操作略快一些。

写操作的选通信号的低电平有效宽度最小是0.16uS或者0.25uS，读操作的选通信号的低电平有效宽度最小是0.25uS或者0.33uS，理想状态下的最大传输速度是800KB/S，在WINDOWS XP SP2环境下使用连续的大数据块进行速度测试，实测传输速度约为：下传510KB/S，上传560KB/S

EPP并口方式应用连接图如下图所示。图中使用非门以及或门进行译码，产生较多的控制信号，如果实际应用只需要较少的输入和输出，那么可以省去这些译码，而直接使用AS#、DS#等控制信号进行简单的实现。

更详细的说明请参考：http://www.usb-i2c-spi.com/cn/rar/USB2XXX_Mamu.pdf。



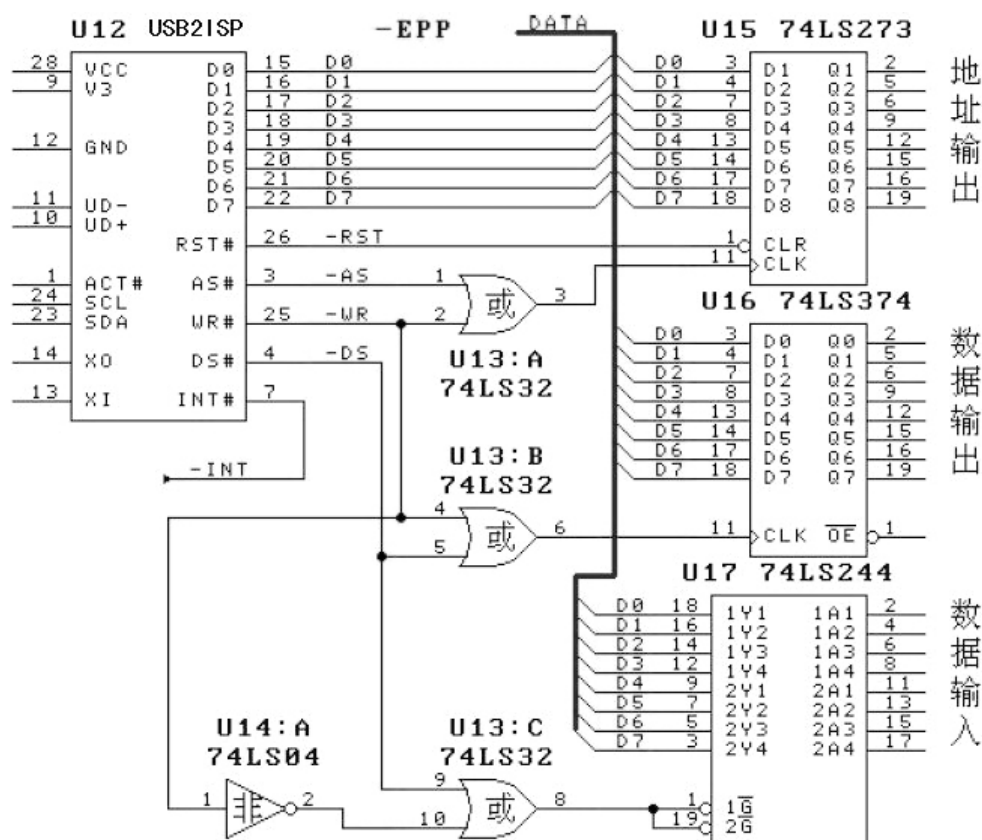


图 9 - USB2ISP 转换成 EPP 并口

7.6. 应用-MEM 并口方式应用

MEM 方式类似于存储器的读写方式，WR#和RD#都是低电平有效的脉冲信号。MEM的实际操作发生于WR#或者RD#有效期间，对计算机端而言，当WR#有效时对外部电路执行写操作，当RD#有效时对外部电路执行读操作。A0 用于指示当前读写操作的地址，例如：将A0 和A0 的反相分别用于两个外部设备的片选，或者将A0=1 时的操作指向外部设备的命令端口，而将A0=0 时的操作指向数据端口。

MEM并口方式应用连接图如下图所示。图中尚未用到A0引脚（AS#引脚），可以实现8位数字信号输入和8位数字信号输出。如果使用A0控制74LS139分别对-READ和-WRITE进行地址片选，那么USB2ISP可以连接两组74LS244+74LS273，从而实现16位数字信号输入和16位数字信号输出。

更详细的说明请参考：http://www.usb-i2c-spi.com/cn/rar/USB2XXX_Mamu.pdf。

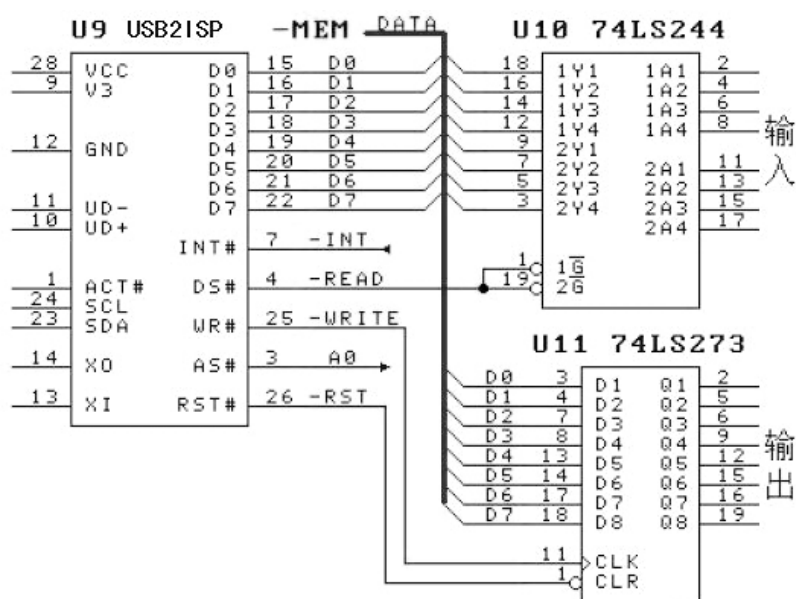


图 10 - USB2ISP 转换成 MEM 并口

8、USB2ISP 常见硬件问题

8.1、关于电容和晶振

USB2ISP可以在XI脚对地加了一个10M的电阻，提高起振的稳定性。

8.2、关于中断设置的说明

不推荐使用中断方式通信，因为USB协议本身的限制，即使采用中断方式对于实时性和USB传输速率并没有太大的改观。

中断调用方式：先定义一个中断程序，

```
USBIO_SetIntRoutine(//设定中断服务程序
```

```
    ULONG iIndex, //指定USB2ISP设备序号
```

```
    mPUSBIO_INT_ROUTINE iIntRoutine); //指定中断服务程序, 为NULL则取消中断服务
```

说明：设置USB2ISP的中断服务程序，iIntRoutine是一个符合mPUSBIO_INT_ROUTINE格式的子程序，当USB2ISP的INT#引脚出现上升沿时，USBIOX.DLL自动调用iIntRoutine，并向其提供一个引脚状态参数，引脚状态参数中，位为1则说明对应的引脚为高电平，位为0则说明对应的引脚为低电平，位7-位0对应USB2ISP的D7-D0引脚，位8对应USB2ISP的ERR#引脚，位9对应USB2ISP的PEMP引脚，位10对应USB2ISP的INT#引脚，位11对应USB2ISP的SLCT引脚。例如：主程序

```
main{
```

```
.....
```

```
    USBIO_OpenDevice(0); //打开设备, 针对0#设备, 如果有多个, 可以计数
```

```
    USBIO_SetIntRoutine(0, myInterruptEvent); //设置中断服务程序
```

```
    ..... 读写数据, 或者在接收到中断服务程序的通知后处理中断
```

```
    USBIO_CloseDevice(0); //用完后关闭设备
```

```
}
```

中断服务程序，当USB2ISP的INT#引脚出现上升沿时，USBIOX.DLL会自动调用该子程序，

```
Void CALLBACK myInterruptEvent(unsigned long PinStatus) {
```

```
    if(PinStatus & mStateBitERR) printf("发生中断时ERR#引脚为高电平");
```

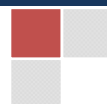
```
    else printf("发生中断时ERR#引脚为低电平");
```

```
    ..... 自己处理或者通知主程序处理
```

```
}
```

8.3、SPI 不可以随便改速率

SPI不可以随便改速率，68K是SPI传输测出的速度。



8.4、I2C 接口上拉电阻

I2C接口上拉电阻大小是47K欧。长距离或者高时钟频率I2C通信的时候，建议在I2C总线SCL和SDA上拉4.7K欧到10K欧电阻。必要时采用专用的I2C总线延长芯片。

8.5、USB2ISP 外围元器件说明

- ✓ 振荡部分：一个12MHz的晶体，两个15pF的振荡电容，引线尽量简短；
- ✓ 电源退耦：一个0.1uF的电源退耦电容104，接于VCC与GND之间，非常必要；
- ✓ 内部电源：一个0.01uF的电容103，接于V3引脚与GND之间，可选，用于降低EMI；
- ✓ 上电复位：一个0.47uF的电容474，接于RSTI引脚与VCC之间，建议，用于可靠复位。

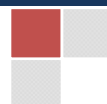
8.6、数据缓冲区是否必须限制 4096

之所以说所谓的限制 4096 缓冲区是出于这样的考虑：由于 WINDOWS 系统的限制，USB 通信实际上是以每 mS 组织“打包”一次数据传输的。即使你把缓冲区开成 1M 的话,那么传输的速度和 4096 字节的传输速度是一样的，那么就没必要提供更多的缓冲区给我们的动态库，而只需要 4096 字节就可以了。

8.7、不能识别 USB 检测

如果第一次插入设备计算机没有提示发现新硬件，请检查硬件，主要检查以下几个方面：

- ✓ (1)-USB 信号线有没有接错 VCC(红),UD-(白),UD+(绿),GND(黑)；
- ✓ (2)-晶振是否起振（如果起振两端的电压应该在 2.5V 左右）；
- ✓ (3)-V3 引脚的电容是否为 103 电容；
- ✓ (4)-USB 线是否为屏蔽线，线的长度不能超过 5M。



9、USB2ISP 驱动

USB2ISP提供Windows和Linux下面的驱动。

9.1、WINDOWS 系统下的驱动

9、1、1. 下载驱动文件

从USBIO Tech.网站www.usb-i2c-spi.com/cn的“在线下载”栏里下载最新版本的驱动程序。
连接地址是：<http://www.usb-i2c-spi.com/cn/down.htm>。

对于USB2ISP、USB2I2C或USB2SPI，选择相应的“开发大礼包”下载。解压缩到本地机器的硬盘里，待用。USB2ISP驱动文件目录如下：

```
├─DRIVER
│   └─DRIVER    //请使用这里的驱动
│       USBIOX.DLL    //动态链接库，可以被VC，VB，CBC，Delphi等调用
│       USBIOX.INF    //
│       USBIOX.SYS    //相应的驱动文件
└─LIB_C
    USBIOX.H          //所有驱动API使用说明，这是一份重要说明文档！！！！
    USBIOX.LIB        //可以被VC，VB，CBC，Delphi等调用的库文件
```

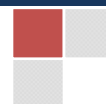
USB2I2C和USB2SPI是USB2ISP的功能简化版本，可以看作是USB2ISP的子集。USB2I2C（SSOP20封装）实现USB转换I2C接口；USB2SPI（SSOP20封装）实现USB转换SPI接口；USB2ISP实现USB转换SPI和I2C接口，同时还提供EPP或MEM并口，也可以当成是GPIO口来使用。

下面以USB2ISP为例，说明USB2XXX系列转换芯片的驱动安装。

9、1、2. 插入USB2ISP_DEV开发板

将USB2ISP_DEV开发板（或者自己设计的USB2XX板子）插入到电脑主板USB接口。当USB2ISP_DEV开发板向外部供电时，最好插入PC机背部的主板USB口。

9、1、3. Windows提示发现新硬件

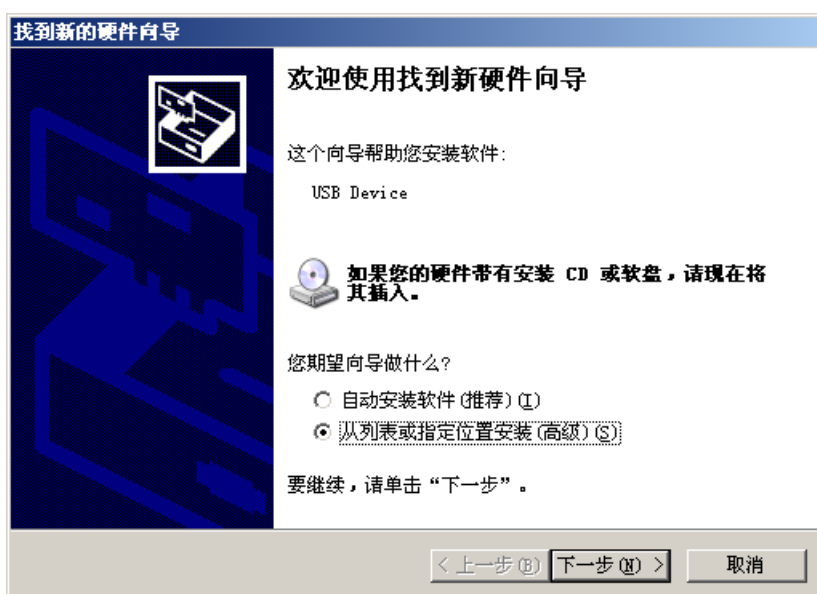




Windows发现了新USB硬件设备

插入 USB2ISP_DEV 开发板后 Windows 提示发现新硬件。

9、1、4. 提示安装驱动

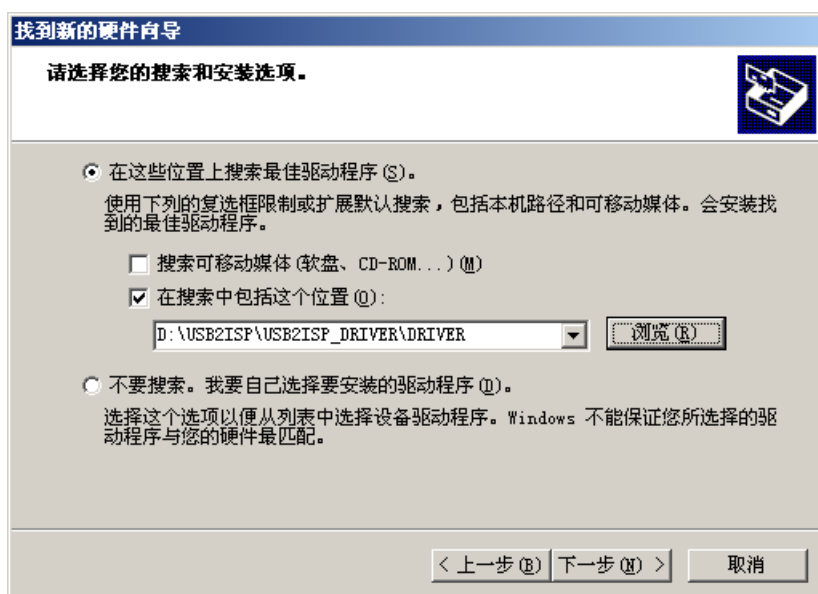


提示安装驱动

选择【从列表或指定位置安装（高级）】选项，然后单击【下一步】按钮。

9、1、5. 指定驱动文件的路径

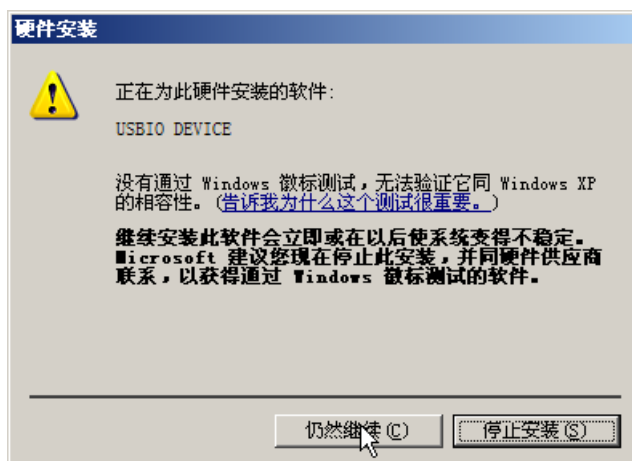
此处需要指定驱动文件的路径。驱动文件就是从网站上下载解压缩后的文件。



9、1、6. 复制文件

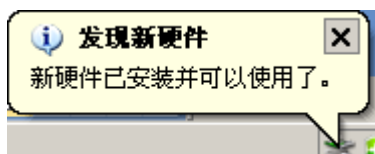


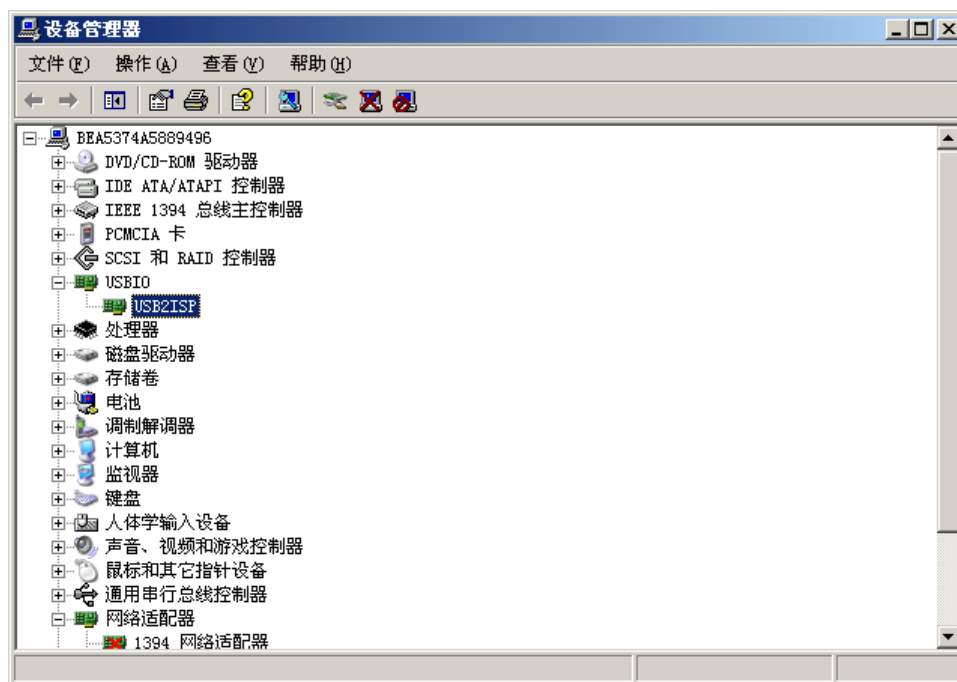
接下来是系统复制驱动的过程。首次安装可能还会提示“没有通过 Windows 徽标测试”，选择【仍然继续】按钮。



微软徽标认证

9、1、7. 安装成功



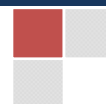


可以通过我的【电脑】→【属性】→【硬件设备管理器】来查看新安装的设备。也可以打开 USB2ISP_DEMO_VBCN.exe , 此时状态来显示

9.2、LINUX 系统下的驱动

USB2ISP提供Linux 2.6内核下面的驱动，需要开源的Libusb支持。

下载地址 :<http://www.usb-i2c-spi.com/cn/download.htm> , 文件名【USB2XXXLinuxDriver】



10、上位 PC 机应用软件开发

在计算机端的Windows操作系统下, USB2ISP的并口驱动程序和动态链接库USBIOX.DLL向应用程序提供了应用层接口, 包括: 设备管理API、并口数据传输API、同步串口数据传输API、中断处理API。有关API参数的说明请参考USBIOX.H(一下各节中说明的API, 均在USBIOX.H有更详细的说明), 主要API如下。

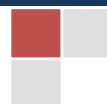
```

├─DRIVER
│   └─DRIVER    //请使用这里的驱动
│       USBIOX.DLL    //动态链接库, 可以被 VC, VB, CBC, Delphi 等调用
│       USBIOX.INF    //
│       USBIOX.SYS    //相应的驱动文件
└─LIB_C
    USBIOX.H    //所有驱动 API 使用说明, 这是一份重要说明文档!!!
    USBIOX.LIB    //可以被 VC, VB, CBC, Delphi 等调用的库文件
  
```

[有关DLL中各个API的使用VB和VC实例请参考USB2ISP评估板资料中的各个源程序及例子。](#)

10.1. 设备管理 API

- ✓ USBIO_OpenDevice(//打开USB2ISP设备, 返回句柄, 出错则无效
 ULONG iIndex); //指定USB2ISP设备序号, 0对应第一个设备
 说明: 将USBIO_作为设备, 使用前必须先打开, 然后才能使用。
- ✓ USBIO_CloseDevice(//关闭USB2ISP设备
 ULONG iIndex); //指定USB2ISP设备序号
 说明: 用完USBIO_后, 或者应用程序退出前, 应该关闭USB2ISP设备。
- ✓ USBIO_SetDeviceNotify(//设定设备事件通知程序
 ULONG iIndex, //指定USB2ISP设备序号, 0对应第一个设备
 PCHAR iDeviceID, //可选参数, 指向字符串, 指定被监控的设备的ID, 字符串以\0终止
 mUSBIO__NOTIFY_ROUTINE iNotifyRoutine); //指定设备事件回调程序
 说明: 用于应用程序监控USB2ISP设备的插拔事件, 确保应用程序随时知道USB设备是否存在, 防止在USB设备拔出后收发数据, 并及时响应USB设备的插入。
- ✓ USBIO_GetStatus(//通过USB2ISP直接输入数据和状态, 类似的API还有USBIO_GetInput
 ULONG iIndex, //指定USB2ISP设备序号
 PULONG iStatus); //指向一个双字单元, 用于保存状态数据
 说明: 获取的状态数据中: 位7-位0对应USB2ISP的D7-D0引脚, 位8对应USB2ISP的ERR#引脚, 位9对应USB2ISP的PEMP引脚, 位10对应USB2ISP的INT#引脚, 位11对应USB2ISP的SLCT引脚, 位13对应USB2ISP的WAIT#引脚, 位14对应USB2ISP的DS#引脚, 位15对应USB2ISP的AS#引脚, 位23对应USB2ISP的SDA引脚。



- ✓ USBIO_SetOutput (//设置USB2ISP的I/O方向, 并通过USB2ISP直接输出数据

ULONG iIndex, //指定USB2ISP设备序号

ULONG iEnable, //数据有效标志

ULONG iSetDirOut, //设置I/O方向, 位清0则对应引脚为输入, 位置1则对应引脚为输出

ULONG iSetDataOut); //输出数据, 如果I/O方向为输出, 那么位数据将通过引脚输出

说明: 谨慎使用该API, 防止修改I/O方向使输入引脚变为输出导致与其它输出引脚之间短路而损坏。上述的I/O方向和输出数据以32位数据表示, 其中: 位7-位0对应USB2ISP的D7-D0引脚, 位8对应USB2ISP的ERR#引脚, 位9对应USB2ISP的PEMP引脚, 位10对应USB2ISP的INT#引脚, 位11对应USB2ISP的SLCT引脚, 位13对应USB2ISP的WAIT#引脚, 位14对应USB2ISP的DS#/READ#引脚, 位15对应USB2ISP的AS#引脚另外, 以下引脚只能输出, 不考虑I/O方向: 位16对应USB2ISP的RESET#引脚, 位17对应USB2ISP的WRITE#引脚, 位18对应USB2ISP的SCL引脚, 位29对应USB2ISP的SDA引脚。

- ✓ USBIO_Set_D5_D0 (//设置USB2ISP的D5-D0引脚的I/O方向, 并通过D5-D0引脚直接输出数据

ULONG iIndex, //指定USB2ISP设备序号

ULONG iSetDirOut, //设置D5-D0各引脚的I/O方向, 清0则引脚为输入, 置1则引脚为输出

ULONG iSetDataOut); //设置D5-D0各引脚的输出数据, 仅当I/O方向为输出时生效

说明: 谨慎使用该API, 防止修改I/O方向使输入引脚变为输出导致与其它输出引脚之间短路而损坏。

10.2. 中断处理 API

- ✓ USBIO_SetIntRoutine (//设定中断服务程序

ULONG iIndex, //指定USB2ISP设备序号

mPUSBIO_INT_ROUTINE iIntRoutine); //指定中断服务程序, 为NULL则取消中断服务

说明: 设置USB2ISP的中断服务程序, iIntRoutine是一个符合mPUSBIO_INT_ROUTINE格式的子程序, 当USB2ISP的INT#引脚出现上升沿时, USBIOX.DLL自动调用iIntRoutine, 并向其提供一个引脚状态参数, 引脚状态参数中, 位为1则说明对应的引脚为高电平, 位为0则说明对应的引脚为低电平, 位7-位0对应USB2ISP的D7-D0引脚, 位8对应USB2ISP的ERR#引脚, 位9对应USB2ISP的PEMP引脚, 位10对应USB2ISP的INT#引脚, 位11对应USB2ISP的SLCT引脚。例如: 主程序

```
main{
.....
USBIO_OpenDevice(0); //打开设备, 针对0#设备, 如果有多个, 可以计数
USBIO_SetIntRoutine(0, myInterruptEvent); //设置中断服务程序
..... 读写数据, 或者在接收到中断服务程序的通知后处理中断
USBIO_CloseDevice(0); //用完后关闭设备
}
```

中断服务程序, 当USB2ISP的INT#引脚出现上升沿时, USBIOX.DLL会自动调用该子程序,

```
Void CALLBACK myInterruptEvent(unsigned long PinStatus) {
```



```

    if(PinStatus & mStateBitERR) printf(“发生中断时ERR#引脚为高电平”);
    else printf(“发生中断时ERR#引脚为低电平”);
    ..... 自己处理或者通知主程序处理
}

```

10.3. 并口数据传输 API

- ✓ USBIO_InitParallel (//复位并初始化并口, RST#输出低电平脉冲

ULONG iIndex, //指定USB2ISP设备序号

ULONG iMode); //指定并口模式: 0为EPP模式, 2为MEM模式, >=256保持当前模式

说明: 在USB2ISP上电时自动初始化并口, 如果需要, 也可以重新初始化并口, 以清除缓冲区。在初始化过程中, RST#引脚会输出100uS左右宽度的低电平脉冲, 用于通知外部设备复位。

- ✓ USBIO_EppReadData (//EPP方式读数据: WR#=1, DS#=0, AS#=1, D0-D7=input

ULONG iIndex, //指定USB2ISP设备序号

ULONG oBuffer, //指向一个足够大的缓冲区, 用于保存读取的数据

PULONG ioLength); //指向长度单元, 输入时为准备读取的长度, 返回后为实际读取的长度

说明: 以EPP时序连续读取数据, 长度为0到4096字节, 例如:

```
ULONG buf[1024];
```

```
ULONG len=1024;
```

```
USBIO_EppReadData(0, buf, &len); //针对0#设备以EPP方式读取1KB数据
```

- ✓ USBIO_EppReadAddr (//EPP方式读地址: WR#=1, DS#=1, AS#=0, D0-D7=input

ULONG iIndex, //指定USB2ISP设备序号

ULONG oBuffer, //指向一个足够大的缓冲区, 用于保存读取的地址数据

PULONG ioLength); //指向长度单元, 输入时为准备读取的长度, 返回后为实际读取的长度

说明: 以EPP时序连续读取地址数据, 长度为0到4096字节, 在标准EPP时序中通常用不到。

- ✓ USBIO_EppWriteData (//EPP方式写数据: WR#=0, DS#=0, AS#=1, D0-D7=output

ULONG iIndex, //指定USB2ISP设备序号

ULONG iBuffer, //指向一个缓冲区, 放置准备写出的数据

PULONG ioLength); //指向长度单元, 输入时为准备写出的长度, 返回后为实际写出的长度

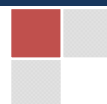
说明: 以EPP时序连续写出数据, 长度为0到4096字节, 例如:

```
ULONG buf[1024];
```

```
ULONG len=1024;
```

```
在buf中放置数据, 准备以EPP方式写出
```

```
USBIO_EppWriteData(0, buf, &len); //针对0#设备以EPP方式写出1KB数据。
```



- ✓ `USBIO_EppWriteAddr` (//EPP方式写地址:WR#=0, DS#=1, AS#=0, D0-D7=output
 `ULONG iIndex`, //指定USB2ISP设备序号
 `ULONG iBuffer`, //指向一个缓冲区, 放置准备写出的地址数据
 `PULONG ioLength`); //指向长度单元, 输入时为准备写出的长度, 返回后为实际写出的长度

说明: 以EPP时序连续写出地址数据, 长度为0到4096字节, 在标准EPP时序中通常只写一个字节的地址。

- ✓ `USBIO_EppSetAddr` (//EPP方式设置地址:WR#=0, DS#=1, AS#=0, D0-D7=output
 `ULONG iIndex`, //指定USB2ISP设备序号
 `ULONG iAddr`); //指定EPP地址以EPP时序输出一个地址, 是`USBIO_EppWriteAddr`的简化。

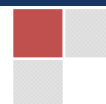
- ✓ `USBIO_MemReadAddr0` (//MEM方式读地址0:WR#=1, DS#/RD#=0, AS#/ADDR=0, D0-D7=input
 `ULONG iIndex`, //指定USB2ISP设备序号
 `ULONG oBuffer`, //指向一个足够大的缓冲区, 用于保存从地址0读取的数据
 `PULONG ioLength`); //指向长度单元, 输入时为准备读取的长度, 返回后为实际读取的长度
- 说明: 以MEM时序连续读取数据, 长度为0到4096字节, 读操作期间ADDR=0, 例如:

```
ULONG buf[1024];
ULONG len=1024;
USBIO_MemReadAddr0(0, buf, &len); //针对0#设备以MEM方式从地址0读取1K数据
```

- ✓ `USBIO_MemReadAddr1` (//MEM方式读地址1:WR#=1, DS#/RD#=0, AS#/ADDR=1, D0-D7=input
 `ULONG iIndex`, //指定USB2ISP设备序号
 `ULONG oBuffer`, //指向一个足够大的缓冲区, 用于保存从地址1读取的数据
 `PULONG ioLength`); //指向长度单元, 输入时为准备读取的长度, 返回后为实际读取的长度
- 说明: 以MEM时序连续读取数据, 长度为0到4096字节, 读操作期间ADDR=1。

- ✓ `USBIO_MemWriteAddr0` (//MEM方式写地址0:WR#=0, DS#/RD#=1, AS#/ADDR=0, D0-D7=output
 `ULONG iIndex`, //指定USB2ISP设备序号
 `ULONG iBuffer`, //指向一个缓冲区, 放置准备向地址0写出的数据
 `PULONG ioLength`); //指向长度单元, 输入时为准备写出的长度, 返回后为实际写出的长度
- 说明: 以MEM时序连续写出数据, 长度为0到4096字节, 写操作期间ADDR=0, 例如:

```
ULONG buf[1024];
ULONG len=1024;
在buf中放置数据, 准备以MEM方式写出
USBIO_MemWriteAddr0(0, buf, &len); //针对0#设备以MEM方式向地址0写出1K数据
```



- ✓ USBIO_MemWriteAddr1 (//MEM方式写地址1:WR#=0, DS#/RD#=1, AS#/ADDR=1, D0-D7=output
 ULONG iIndex, //指定USB2ISP设备序号
 ULONG iBuffer, //指向一个缓冲区, 放置准备向地址1写出的数据
 PULONG ioLength); //指向长度单元, 输入时为准备写出的长度, 返回后为实际写出的长度
 说明: 以MEM时序连续写出数据, 长度为0到4096字节, 写操作期间ADDR=1

10.4. 同步串口数据传输 API

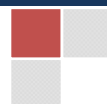
- ✓ USBIO_ReadI2C (//从两线串口读取一个字节数据, 仅适用于7位地址的设备
 ULONG iIndex, //指定USB2ISP设备序号
 ULONG iDevice, //低7位指定设备地址
 ULONG iAddr, //指定数据单元的地址
 PULONG oByte); //指向一个字节单元, 用于保存读取的字节数据
 说明: 从两线串口读取一个字节数据。仅适用于7位地址的设备, 不支持带从地址的I2C设备。

- ✓ USBIO_WriteI2C (//向两线串口写入一个字节数据, 仅适用于7位地址的设备
 ULONG iIndex, //指定USB2ISP设备序号
 ULONG iDevice, //低7位指定设备地址
 ULONG iAddr, //指定数据单元的地址
 ULONG iByte); //待写入的字节数
 说明: 从两线串口读取一个字节数据。仅适用于7位地址的设备, 不支持带从地址的I2C设备。

- ✓ USBIO_WriteRead (//执行数据流命令, 先输出再输入
 ULONG iIndex, //指定USB2ISP设备序号
 ULONG iWriteLength, //写长度, 准备写出的长度
 ULONG iWriteBuffer, //指向一个缓冲区, 放置准备写出的数据
 ULONG iReadStep, //准备读取的单个块的长度, 总长度为(iReadStep*iReadTimes)
 ULONG iReadTimes, //准备读取的次数
 PULONG oReadLength, //指向长度单元, 返回后为实际读取的长度
 ULONG oReadBuffer); //指向一个足够大的缓冲区, 用于保存读取的数据
 说明: 先输出数据再输入数据, 执行数据流命令, 适用于同步串口等。

- ✓ USBIO_SetStream (//设置同步串口流模式
 ULONG iIndex, //指定USB2ISP设备序号
 ULONG iMode); //指定模式, 见下面的说明
 说明: IMODE的位1位0: I2C速度/SCL频率, 00=低速20KHZ, 01=标准100KHZ, 10=快速400KHZ, 11=高速750KHZ//位2: SPI的I/O数/I/O引脚, 0=单入单出(4线接口), 1=双入双出(5线接口)//位7: SPI字节中的位顺序, 0=低位在前, 1=高位在前//其它保留, 必须为0。

- ✓ USBIO_StreamI2C (//处理两线串口的数据流, 适用于所有两线串口的设备



```

ULONG iIndex, //指定USB2ISP设备序号
ULONG iWriteLength, //准备写出的数据字节数
ULONG iWriteBuffer, //指向缓冲区, 放置准备写出的数据, 首字节是设备地址及读写位
ULONG iReadLength, //准备读取的数据字节数
ULONG oReadBuffer); //指向缓冲区, 返回后是读入的数据对两线串口设备进行操作。

```

例如, 从24C256中3200H开始的地址读出256字节的数据:

```

ULONG OutBuf[5], InBuf[300]; //待写数据缓冲区, 读出数据缓冲区
OutBuf[0]=0xA1;
OutBuf[1]=0x32;
OutBuf[2]=0x00; //待写数据: 设备地址及单元地址
USBIO_StreamI2C(0, 3, OutBuf, 256, InBuf); //针对0#设备处理两线串口的数据流

```

- ✓ USBIO_ReadEEPROM(//从EEPROM中读取数据块, 速度约56K字节

```

ULONG iIndex, //指定USB2ISP设备序号
EEPROM_TYPE iEepromID, //指定EEPROM型号
ULONG iAddr, //指定数据单元的地址
ULONG iLength, //准备读取的数据字节数
PULONG oBuffer); //指向一个缓冲区, 返回后是读入的数据

```

说明: 读EEPROM的API支持从24C01到24C16和从24C32到24C4096的各种型号的EEPROM存储器。

- ✓ USBIO_WriteEEPROM(//向EEPROM中写入数据块

```

ULONG iIndex, //指定USB2ISP设备序号
EEPROM_TYPE iEepromID, //指定EEPROM型号
ULONG iAddr, //指定数据单元的地址
ULONG iLength, //准备写出的数据字节数
PULONG iBuffer); //指向一个缓冲区, 放置准备写出的数据

```

说明: 写EEPROM的API支持从24C01到24C16和从24C32到24C4096的各种型号的EEPROM存储器。

- ✓ USBIO_StreamSPI4(//处理SPI数据流, 4线接口, 速度约68K字节

```

ULONG iIndex, //指定USB2ISP设备序号
ULONG iChipSelect, //片选控制, 位7为0则忽略片选控制, 位7为1则参数有效
ULONG iLength, //准备传输的数据字节数
ULONG ioBuffer); //缓冲区, 放置准备从DOUT (MOSI) 写出的数据, 返回后是从DIN (MISO) 读入的数据

```

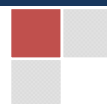
说明: 主SPI接口MODELO模式时序: DCK时钟输出, 默认为低, DOUT (MOSI) 在时钟上升沿之前输出, DIN (MISO) 在时钟下降沿之后输入。SPI数据流速度约68K字节/S, SCK时钟频率约为568KHZ, 这个时钟的频率是固定的, 不可调。

- ✓ USBIO_StreamSPI5(//处理SPI数据流, 5线接口

```

ULONG iIndex, //指定USB2ISP设备序号
ULONG iChipSelect, //片选控制, 位7为0则忽略片选控制, 位7为1则参数有效: 位1位0为00/01/10
分别选择D0/D1/D2引脚作为低电平有效片选

```



ULONG iLength, //准备传输的数据字节数

ULONG ioBuffer, //指向一个缓冲区, 放置准备从DOUT (MOSI) 写出的数据, 返回后是从DIN (MISO) 读入的数据

ULONG ioBuffer2); //指向第二个缓冲区, 放置准备从DOUT (MOSI) 2写出的数据, 返回后是从DIN (MISO) 2读入的数据

说明: 时钟线为DCK/D3引脚, 输出数据线为DOUT (MOSI) /D5和DOUT (MOSI) 2/D4引脚, 输入数据线为DIN (MISO) /D7和DIN (MISO) 2/D6引脚, 片选线为D0/D1/D2, 速度约30K字节*

5线制SPI时序: DCK/D3引脚为时钟输出, 默认为低电平, DOUT (MOSI) /D5和DOUT (MOSI) 2/D4引脚在时钟上升沿之前输出, DIN (MISO) /D7和DIN (MISO) 2/D6引脚在时钟下降沿之后输入。

11、封装尺寸

USB2ISP采用SOP28封装, PCB尺寸如下图。左图以mil为单位, 右图以mm为单位。

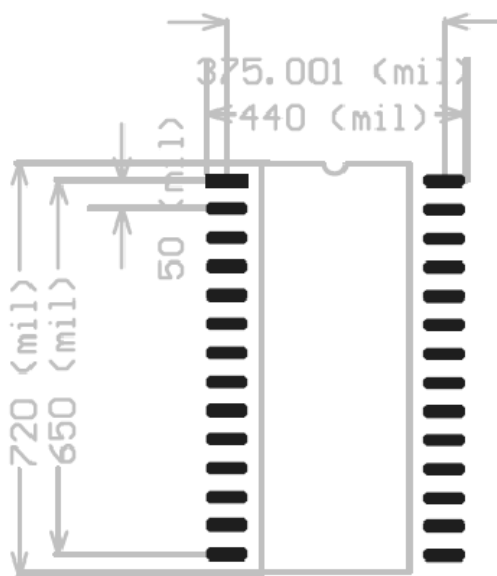
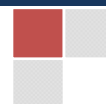


图 11 - USB2ISP 封装图 (MIL)



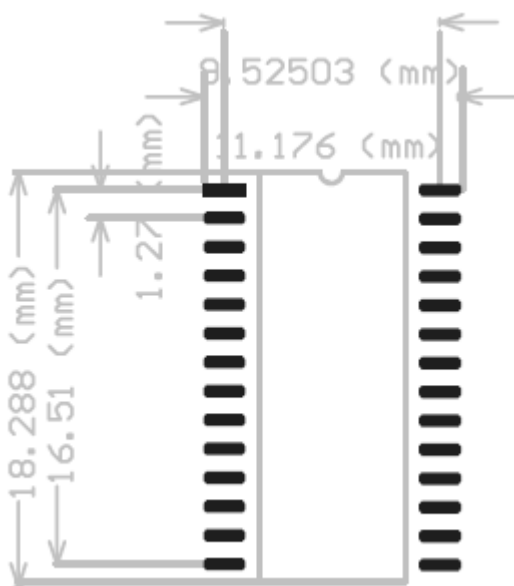


图 12 - USB2ISP 封装图 (MM)

版权

2007年6月版，版权属USBIO科技发展有限公司所有，未经USBIO科技发展有限公司事先的书面允许，本出版物的任何部分不得被翻版、传播。

本手册中所包含的内容发生变更时，恕不另行通知。

