第35卷 第10期 计 算 机 工 程 2009年5月
Vol.35 No.10 Computer Engineering May 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)10—0246—02 文献标识码：A 中图分类号：TP311.52

基于USB设备的COP复位处理方法

曹振华，王宜怀，葛 强
(苏州大学计算机科学与技术学院，苏州 215006)

摘 要：嵌入式系统中引入COP能保证系统能从故障点自动恢复正常运行。COP复位后采用简单的内存恢复方法即可使系统恢复正常运行，但不适用于带有USB等特殊模块的系统。研究USB模块运行机制，设计带有USB模块的数据采集系统，提出一种基于寄存器恢复的COP复位处理方法，该方法能有效解决带有类似特殊模块的系统的COP复位问题。

关键词：通用串行总线协议1.0；系统正常操作监视模块；JB8单片机

COP Reset Processing Method Based on USB Device

CAO Zhen-hua, WANG Yi-huai, GE Qiang
(School of Computer Science and Technology, Soochow University, Suzhou 215006)

【Abstract】 The introduction of Computer Operating Properly(COP) in embedded system can guarantee the normal operation of system, and automatically come back to the normal operation from failure. It does not require special treatment or only to use simple memory recovery after COP reset will be able to bring the system back to the normal running. If system includes some special module such as USB module, the above methods can not work. Take a data acquisition system as a example, the failure mechanism about USB module are discussed, and a countermeasures based on register recovery is given, which can effectively rescue system's similar paralys after COP reset.

【Key words】 USB1.0; Computer Operating Properly(COP); JB8 MCU

COP(Computer Operating Properly)，俗称看门狗，是一种能保证系统从故障恢复点自动转入到正常运行轨道上来的机制[1]。理论上只要单片机方程序都是自己的主动行为，系统经过COP复位重新初始化后都可以正常工作。但在含有特殊模块(如USB模块)的系统中，由于在模块中含有不能完全由自己设定的参数和信息，很难保证系统在COP复位后仍能正常运行，因此，在COP复位前后一定要对这些特殊模块的寄存器进行专门的保护和设置。本文以1个包含JB8 USB模块的数据采集系统为例，分析USB模块在COP复位后不能正常工作的原因，并给出解决类似问题的思路和方法。

1 JB8 USB模块及COP模块

JB8单片机是飞思卡尔半导体公司生产的一款高性能低价位单片机。除了内部集成USB1.0模块、COP模块以外，片内还集成了少量通用模块和I/O资源，便于用户的快速开发[2-3]。JB8单片机被普遍用于数据传输量较大的嵌入式产品设计中。USB技术的应用日渐增多，已得到了450多家技术公司组成的技术联盟的支持[4]。它的高速性、稳定性、支持即插即用、热插拔等优良特性，以及通信领域中依赖的串口的市场淡出趋势，无疑使USB会在不远的将来成为嵌入式领域的标准接口之一。

本文设计一种基于JB8单片机的数据采集系统，利用USB接口与高端PC进行数据通信。在硬件设计中，JB8单片机通过USB电缆由PC机提供工作电源，因此当USB设备进行插拔时，整个数据采集系统会掉电。由于没有考虑到USB模块的特殊性，因此引入COP功能后未对COP复位进行特殊处理。系统运行中有时会提示读取不到USB设备描述符和设备号，无法再继续通信，操作系统并不提示发现无法识别的USB设备，但冷复位之后系统会再次正常运行。

2 USB模块在COP复位后出现的故障分析

为解决上述问题，首先要理解USB工作过程中，与USB模块描述符等信息配置相关的高低两端之间的交互过程。

2.1 USB模块信息配置过程

系统在COP复位后不能正常工作，但冷复位后能正常工作，而2种复位的最大区别在于COP复位系统不掉电，并且内存信息不会丢失。USB设备插到PC机USB总线上以后，相当于冷复位，高端PC机依靠其操作系统的定时检测机制，能及时发现有USB设备已经插入到了USB总线上[5]，然后由高端PC操作系统主动发起USB设备的设备列举，此列举是通过多次USB中断与低端MCU交互完成的，PC端操作系统首先通过硬件默认的地址0x00与低端MCU交互，索取USB设备的设备描述符、设备号等相关信息，并在设置的过程中分配给每个已经检测到的USB设备一个独立的地址来区分不同的USB设备。低端MCU方接收到高端分配的这个地址后，会将自己的默认的7位地址0x00改成这个新地址，保存在寄存器中，高端PC操作系统也会把这些信息保存到自己的设备表中，高低端之间即可进行数据通信。

PC机的USB总线上没有USB设备进行插拔时，操作系统的定时检测机制检测不到所连接设备的变化，不会使自己重新进行设备列举，仍使用设备表中的设备信息(包括设备地

基金项目：国家科技部技术创新基金资助项目(04C2262232200503)
作者简介：曹振华(1980－)，男，硕士研究生，主研方向：嵌入式应用技术；王宜怀，博士；葛 强，硕士研究生
收稿日期：2008-09-17 **E-mail：**luoyujiangnan@126.com

址等)进行通信。

2.2 COP 复位后出现故障的原因分析

通过对寄存器配置过程的分析可知，系统进行冷复位时，高端 PC 的操作系统能检测到设备变动并进行设备列举，在此后的数据通信中高低端都是用的最新的设备信息进行通信的，所以系统能正常工作[6]。系统发生 COP 复位时，PC 操作系统不会重新进行设备列举，操作系统维护的设备表也就得不到更新。因此，操作系统与低端 MCU 数据通信时，仍用这张没有更新过的表中的参数进行通信。但由于低端 MCU 已经进行了 COP 复位，MCU 中 USB 模块的参数(如地址寄存器等)都已经变成了硬件默认的参数了(USB 模块相关寄存器请参考 JB8 芯片手册)。因此高低端通信用到的参数就出现了不一致的的现象。高端向某一低端设备索取设备描述符等信息时，由于 PC 机操作系统的设备表中的这个设备信息已没有低端设备信息与其对应，因此无法获取相关的信息，从而出现上文提到的未能获得设备描述符等提示错误。但由于不是在设备列举时发现的错误，因此并不报告无法识别的 USB 设备。

3 USB 模块在 COP 复位后的处理方法

利用热复位内存中的数据不会丢失(除了被覆盖的部分)这种特性，可以把系统正常运行时 USB 模块相关的数据保存在热复位后不会被程序覆盖的特定内存区域内，热复位后，再将记录在内存中的相关信息回复到相应的寄存器中，从而解决热复位前后参数不一致的问题。

3.1 寄存器数据的保存

参考 JB8 芯片手册可知，USB 模块有 10 个相关寄存器，其中状态寄存器不能人为干预，数据寄存器与通信参数无关，因此只需要记录和恢复其中的 8 个寄存器的值即可保证一致性。在内存中开辟 8 个字节的临时变量结构体保存这些寄存器在正常工作时的值，结构体定义代码如下：

```
struct USB_reg
{INT8U UCR0_reg;//保存 UCR0 的值
INT8U UCR1_reg; //保存 UCR1 的值
…
INT8U UIR2_reg;//保存 UIR2 的值
INT8U UADDR_reg;//保存 UADDR 的值};
```

USB 模块相关信息的设置和修改的地方有 2 个：(1)进行 USB 模块初始化时设置的部分基本信息。(2)主程序初始化完毕并开放总中断后，在相应高端 PC 操作系统的设备列举的中断处理程序中设置的信息。按时间的先后可推断，在高端 PC 操作系统的设备列举完成时，即低端 MCU 退出中断时的寄存器的值就是能保证系统能正常工作的数据。因此，可将保存寄存器值的操作放到中断处理程序中“ asm(“cli”); ”语句前，程序代码如下：

```
//将 8 个 USB 寄存器值保存到 USB_reg 中，此语句放在与高端
//交互的 USB 中断处理函数中。
USB_reg->UCR0_reg=UCR0;//保存 UCR0 的值到 USB_reg 中
USB_reg->UCR1_reg=UCR1;//保存 UCR1 的值到 USB_reg 中
…
USB_reg->UIR2_reg=UIR2;//保存 UIR2 的值到 USB_reg 中
USB_reg->UADDR_reg=UADDR;// 保存 UADDR 的值到
//USB_reg 中
asm(“cli”);
```

3.2 寄存器数据的恢复

为正确处理 COP 复位，可在主程序之前加一个 COP 复位判断和热复位处理程序，加入了 COP 复位处理程序后的软件流程如图 1 所示。

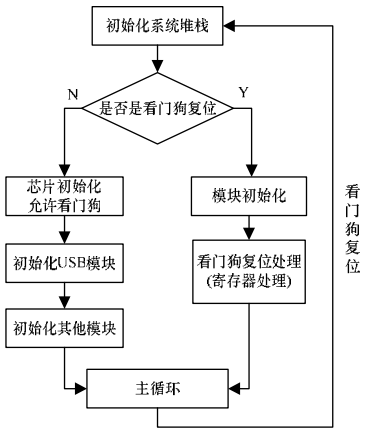

图 1 加 COP 复位处理程序后的软件流程

可将恢复 USB 寄存器值的语句封装成一个子函数 *USB_Recover*，在主程序中如果判断是 COP 复位，除了进行一般模块的初始化以外，在系统初始化过程中开放总中断之前，只要调用 USB_Recover 程序恢复 USB 模块寄存器的值，即可保证 USB 模块 COP 在复位前后参数的一致性。*USB_Recover* 子函数实现代码如下：

```
//功能：恢复 8 个 USB 寄存器值
void USB_Recover()
{UCR0=USB_reg->UCR0_reg; //恢复 UCR0 的值
UCR1=USB_reg->UCR1_reg; //恢复 UCR1 的值
…
UIR2 =USB_reg->UIR2_reg; //恢复 UIR2 的值
UADDR=USB_reg->UADDR_reg; //恢复 UADDR 的值}
```

3.3 处理方法的改进

冷复位以后，如果在初始化完成之后也调用 *USB_Recover* 子函数，内存中的变量值未被赋值，可能是个随机数，程序可能会将一些错误值写入到寄存器中去，但由于是冷复位，随后高端操作系统发起的设备列举会重新给寄存器赋予正确的值，因此系统仍能正常进行 USB 设备识别和通信。这样为保持冷热复位时程序的一致性，在程序中就可以不再判断其是否为热复位，在开放总中断前一句的位置都调用这个恢复寄存器值的 *USB_ Recover* 子函数，即能保证无论发生何种复位，系统都能进行正常的设备检测和正常通信。系统最终的软件流程如图 2 所示。

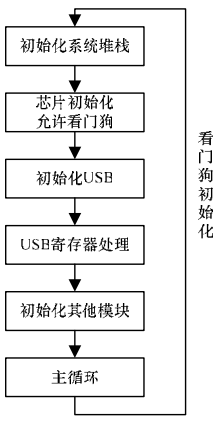

图 2 系统最终的软件流程

(下转第 250 页)