

# 单片机系统的低功耗设计策略

■ 清华大学 陈萌萌 邵贝贝

## 摘要

嵌入式系统的低功耗设计需要全面分析各方面因素,统筹规划。在设计之初,各个因素往往是相互制约、相互影响的,一个降低系统功耗的措施有时会带来其他方面的“负效应”。因此,降低系统整体功耗,需要仔细分析和计算。本文从硬件和应用软件设计两个方面,阐述一个以单片机为核心的嵌入式系统低功耗设计时所需考虑的一些问题。

关键词 低功耗设计 硬件设计 应用软件设计 低功耗模式

在嵌入式应用中,系统的功耗越来越受到人们的重视,这一点对于需要电池供电的便携式系统尤其明显。降低系统功耗,延长电池的寿命,就是降低系统的运行成本。对于以单片机为核心的嵌入式应用,系统功耗的最小化需要从软、硬件设计两方面入手。

随着越来越多的嵌入式应用使用了实时操作系统,如何在操作系统层面上降低系统功耗也成为一个问题。限于篇幅,本文仅从硬件设计和应用软件设计两个方面讨论。

## 1 硬件设计

选用具有低功耗特性的单片机可以大大降低系统功耗。可以从供电电压、单片机内部结构设计、系统时钟设计和低功耗模式等几方面考察一款单片机的低功耗特性。

### 1.1 选用尽量简单的 CPU 内核

在选择 CPU 内核时切忌一味追求性能。8 位机够用,就没有必要选用 16 位机,选择的原则应该是“够用就好”。现在单片机的运行速度越来越快,但性能的提升往往带来功耗的增加。一个复杂的 CPU 集成度高、功能强,但片内晶体管多,总漏电流大,即使进入 STOP 状态,漏电流也变得不可忽视;而简单的 CPU 内核不仅功耗低,成本也低。

### 1.2 选择低电压供电的系统

降低单片机的供电电压可以有效地降低其功耗。当前,单片机从与 TTL 兼容的 5 V 供电降低到 3.3 V、3 V、2 V 乃至 1.8 V 供电。供电电压降下来,要归功于半导体工艺的发展。从原来的 3  $\mu\text{m}$  工艺到现在的 0.25、0.18、

0.13  $\mu\text{m}$  工艺, CMOS 电路的门限电平阈值不断降低。低电压供电可以大大降低系统的工作电流,但是由于晶体管的尺寸不断减小,管子的漏电流有增大的趋势,这也是对降低功耗不利的一个方面。

目前,单片机系统的电源电压仍以 5 V 为主,而过去 5 年中,3 V 供电的单片机系统数量增加了 1 倍,2 V 供电的系统也在不断增加。再过五年,低电压供电的单片机数量可能会超过 5 V 电压供电的单片机。如此看来,供电电压降低将是未来单片机发展的一个重要趋势。

### 1.3 选择带有低功耗模式的系统

低功耗模式指的是系统的等待和停止模式。处于这类模式下的单片机功耗将大大小于运行模式下的功耗。过去传统的单片机,在运行模式下有 wait 和 stop 两条指令,可以使单片机进入等待或停止状态,以达到省电的目的。

等待模式下, CPU 停止工作,但系统时钟并不停止,单片机的外围 I/O 模块也不停止工作;系统功耗一般降低有限,相当于工作模式的 50%~70%。

停止模式下,系统时钟也将停止,由外部事件中断重新启动时钟系统时钟,进而唤醒 CPU 继续工作, CPU 消耗电流可降到  $\mu\text{A}$  级。在停止模式下, CPU 本身实际上已经不消耗什么电流,要想进一步减小系统功耗,就要尽量将单片机的各个 I/O 模块关掉。随着 I/O 模块的逐个关闭,系统的功耗越来越小,进入停止模式的深度也越来越深。进入深度停止模式无异于关机,这时的单片机耗电可以小于 20 nA。其中特别要提示的是,片内 RAM 停止供电后, RAM 中存储的数据会丢失,也就是说,唤醒 CPU 后

要重新对系统作初始化。因此在让系统进入深度停止状态前,要将重要系统参数保存在非易失性存储器中,如EEPROM中。深度停止模式关掉了所有的I/O,可能的唤醒方式也很有限,一般只能是复位或IRQ中断等。

保留的I/O模块越多,系统允许的唤醒中断源也就越多。单片机的功耗将根据保留唤醒方式的不同,降至 $1\mu\text{A}$ 至几十 $\mu\text{A}$ 之间。例如,用户可以保留外部键盘中断,保留异步串行口(SCI)接收数据中断等来唤醒CPU。保留的唤醒方式越多,系统耗电也就多一些。其他可能的唤醒方式还有实时钟唤醒、看门狗唤醒等。停机状态较浅的情况下,外部晶振电路还是工作的。

图1以Freescale的HCS08单片机为例,给出不同运行模式下的系统功耗。HCS08是8位单片机,有多个系列,各系列I/O模块数目有所不同,但低功耗模式下的电流消耗大致相同。

以R系列单片机为例:在室温( $25^{\circ}\text{C}$ )下,不包括I/O口的负载,以2V供电,将可编程锁相环时钟设为16MHz(总线时钟8MHz),典型电流值为2.6mA,当温度升高到 $85^{\circ}\text{C}$ 时,供电电流也升高到3.6mA;而采用3V供电,这一组数据升高至3.8mA和4.8mA。用2V供电,直接使用外部晶振2MHz(总线时钟1MHz)时,典型运行电流降至 $450\mu\text{A}$ 。在等待状态下,因时钟并没有停止,耗电情况和时钟频率有很大关系,节省的功耗有限;而进入轻度停止(stop3),以外部中断唤醒,电流消耗在 $0.5\mu\text{A}$ 左右。在中度停止态(stop2),功耗可进一步降低。使用内部1kHz的时钟,保持1个运行的时钟,周期性唤醒CPU,所增加的电流约为 $0.3\mu\text{A}$ 。在深度停止态(stop1),RAM的数据也不再保留,只能通过外部复位重启系统,此时的电流消耗可降到20nA。以上数据都是在室温下测量所得。当环境温度升高到 $85^{\circ}\text{C}$ 时,电流消耗可能增

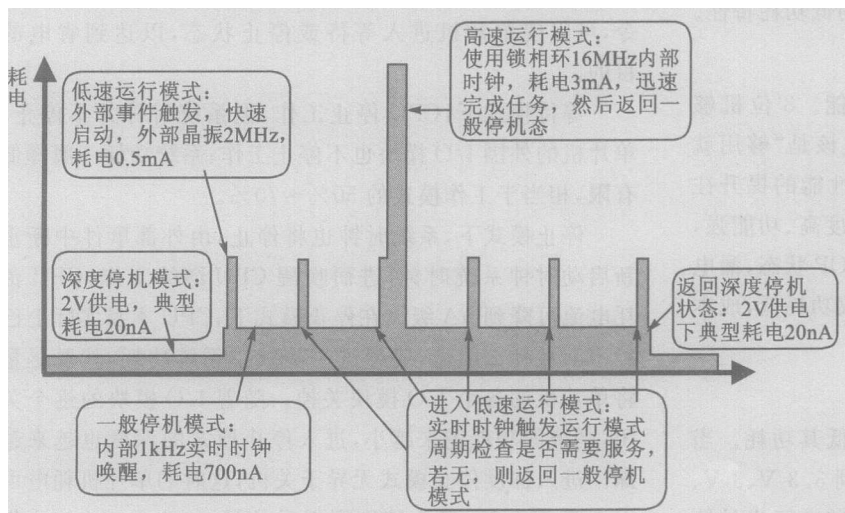


图1 HCS08 单片机各模式下的耗电

加3~5倍。

## 1.4 选择合适的时钟方案

时钟的选择对于系统功耗相当敏感,设计者需要注意两个方面的问题:

第一是系统总线频率应当尽量低。单片机内部的总电流消耗可分为两部分——运行电流和漏电流。理想的CMOS开关电路,在保持输出状态不变时,是不消耗功率的。例如,典型的CMOS反相器电路,如图2所示,当输入端为零时,输出端为1,P晶体管导通,N晶体管截止,没有电流流过。而实际上,由于N晶体管存在一定漏电流,且随集成度提高,管基越薄,漏电流会加大。温度升高,CMOS翻转阈电压会降低,而漏电流则随环境温度的增高变大。在单片机运行时,开关电路不断由“1”变“0”、由“0”变“1”,消耗的功率是由单片机运行引起的,我们称之为“运行电流”。如图2所示,

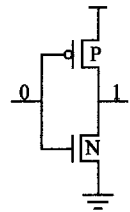


图2 典型的CMOS反相器

在两只晶体管互相变换导通、截止状态时,由于两只管子的开关延迟时间不可能完全一致,在某一瞬间会有两只管子同时导通的情况,此时电源到地之间会有一个瞬间较大的电流,这是单片机

运行电流的主要来源。可以看出,运行电流几乎是和单片机的时钟频率成正比的,因此尽量降低系统时钟的运行频率可以有效地降低系统功耗。

第二是时钟方案,也就是是否使用锁相环、使用外部晶振还是内部晶振等问题。新一代的单片机,如飞思卡尔的HCS08系列单片机,片内带有内部晶振,可以直接作为时钟源。使用片内晶振的优点是可以省掉片外晶振,降低系统的硬件成本;缺点是片内晶振的精度不高(误差一般在25%左右,即使校准之后也可能有2%的相对误差),而且会增加系统的功耗。

现代单片机普遍采用锁相环技术,使单片机的时钟频率可由程序控制。锁相环允许用户在片外使用频率较低的晶振,可以大大地减小板级噪声;而且,由于时钟频率可由程序控制,系统时钟可以在一个很宽的范围调整,总线频率往往能升得很高。但是,使用锁相环也会带来额外的功率消耗。

单就时钟方案来讲,使用外部晶振且不使用锁相环是功率消耗最小的一种。

## 2 应用软件方面的考虑

之所以使用“应用软件”的说法,是为

了区分于“系统软件”或者“实时操作系统”。软件对于一个低功耗系统的重要性常常被人们忽略。一个重要的原因是,软件上的缺陷并不像硬件那样容易发现,同时也没有一个严格的标准来判断一个软件的低功耗特性。尽管如此,设计者仍需尽量将应用的低功耗特性反映在软件中,以避免那些“看不见”的功耗损失。

## 2.1 用“中断”代替“查询”

一个程序使用中断方式还是查询方式对于一些简单的应用并不那么重要,但在其低功耗特性上却相去甚远。使用中断方式,CPU可以什么都不做,甚至可以进入等待模式或停止模式;而查询方式下,CPU必须不停地访问 I/O 寄存器,这会带来很多额外的功耗。

## 2.2 用“宏”代替“子程序”

程序员必须清楚,读 RAM 会比读 Flash 带来更大的功耗。正是因为如此,低功耗性能突出的 ARM 在 CPU 设计上仅允许一次子程序调用。因为 CPU 进入子程序时,会首先将当前 CPU 寄存器推入堆栈(RAM),在离开时又将 CPU 寄存器弹出堆栈,这样至少带来两次对 RAM 的操作。因此,程序员可以考虑用宏定义来代替子程序调用。对于程序员,调用一个子程序还是一个宏在程序写法上并没有什么不同,但宏会在编译时展开,CPU 只是顺序执行指令,避免了调用子程序。唯一的问题似乎是代码量的增加。目前,单片机的片内 Flash 越来越大,对于一些不在乎程序代码量大一些的应用,这种做法无疑会降低系统的功耗。

## 2.3 尽量减少 CPU 的运算量

减少 CPU 运算的工作可以从很多方面入手:将一些运算的结果预先算好,放在 Flash 中,用查表的方法替代实时的计算,减少 CPU 的运算工作量,可以有效地降低 CPU 的功耗(很多单片机都有快速有效的查表指令和寻址方式,用以优化查表算法);不可避免的实时计算,算到精度够了就结束,避免“过度”的计算;尽量使用短的数据类型,例如,尽量使用字符型的 8 位数据替代 16 位的整型数据,尽量使用分数运算而避免浮点数运算等。

## 2.4 让 I/O 模块间歇运行

不用的 I/O 模块或间歇使用的 I/O 模块要及时关掉,以节省电能。RS-232 的驱动需要相当的功率,可以用单片机的一个 I/O 引脚来控制,在不需通信时,将驱动关掉。不用的 I/O 引脚要设置成输出或设置成输入,用上拉电阻拉高。因为如果引脚没有初始化,可能会增大单片机的漏电流。特别要注意有些简单封装的单片机没有把个别 I/O 引脚引出来,对这些看不见的 I/O 引脚也不应忘记初始化。

## 3 结论

一个成功的低功耗设计应该是硬件设计和软件设计的结合。从硬件设计开始,就应该充分意识到一个低功耗应用的特性,选择一款合适的单片机,通过对其特性的了解,设计系统方案;在软件设计上,要考虑到低功耗编程的特殊性,并尽量使用单片机的低功耗模式。

限于篇幅,仅仅讨论了低功耗设计中的一些常见问题,更多的问题只能靠设计者去实际分析和解决了。

### 参考文献

- 1 刘慧银,等. Motorola 微控制器 MC68HC08 原理及其嵌入式应用,北京:清华大学出版社,2001
- 2 邵贝贝. 单片机嵌入式应用的在线开发方法. 北京:清华大学出版社,2004
- 3 Donnie Garcia, Scott Pape. MC9S08GB/GT Low-Power Modes. Freescale Semiconductor, Rev2. 2004
- 4 MC9S08GB/GT Data Sheet. Freescale Semiconductor, Rev. 2.2, 2004
- 5 HCS08 Family Reference Manual. Freescale Semiconductor, 2003
- 6 Scott Pape. HC08 to HCS08 Transition. Freescale Semiconductor, 2004
- 7 Bill Lucas, Scott Pape. Configuring the System and Peripheral Clocks in the MC9S08GB/GT. Freescale Semiconductor, 2003
- 8 Scott Pape. S08 in Low Power Devices. Freescale Technology Forum, 2005

(收稿日期:2005-09-26)

## 请作者及时确认是否收到稿费与赠阅期刊

本刊自创刊以来,对于已录用的稿件,一经发表,都会通过邮局寄给第一作者稿费与样刊。稿费于文章发表后一个月左右寄出,登载有作者文章的期刊(两本)于当月初寄出。

经查,有不少作者因地址变更或其他原因,寄出的稿费及赠刊均被邮局退回。所有退回的稿费均被妥善保存。希望未收到稿费的作者与我社联系,以便再次奉上。

今后,请作者及时确认是否收到稿费与赠阅期刊。通信地址变更时,请及时通知本刊编辑部。

本刊编辑部