第 33 卷 第 6 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.6 Computer Engineering March 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)06—0259—03 文献标识码：A 中图分类号：TP316

μC/OS-II 在总线式数据采集中的应用

金 纯[1,2]，吴小波[1]，王时龙[1]，陈 峰[1]
(1. 重庆大学软件学院，重庆 400039；2. 重庆金瓯科技发展有限责任公司，重庆 400039)

摘 要：介绍一种公开源代码的实时操作系统 μC/OS-II，并对它的实时性能进行简单的分析。讨论了 μC/OS-II 在实际开发应用中应注意的几个问题，并通过实例论述实时操作系统在总线式数据采集中的应用。
关键词：μC/OS-II；实时性能；数据采集

μC/OS-II Application in Figures Gather of Bus

JIN Chun [1,2], WU Xiaobo[1], WANG Shilong[1], CHEN Feng[1]
(1. College of Software, Chongqing University, Chongqing 400039;
2. Chongqing Jinou Science & Technology Development Co., Ltd., Chongqing 400039)

【Abstract】 The thesis introduces a kind of real time operating systems(RTOS) of open code: μC/OS-II. It analyzes the real time function concisely. It discusses several important problems about μC/OS-II in actual developing and application, on the other hand, elucidates that the real time operating systems applied in figures gather by examples.
【Key words】 μC/OS-II; Real time function; Figures gather

μC/OS-II 是一个源代码开放的实时操作系统，可移植、可固化（嵌入到产品中成为产品的一部分）、可裁减，属于占先式实时内核。执行时间可确定（即函数调用与服务的时间是可知的，不依赖于应用程序的多少），支持现有大多数型号的 8 位、16 位、32 位 MCU/MPU，已被广泛应用于交换机、路由器、过程控制、汽车业、办公自动化、计算机外设以及民用消费类产品等，具有稳定的可靠性。把 μC/OC-II 应用在总线式数据采集系统中，可使该系统比以往的前后台系统能够更加稳定的工作，而且在一定程度上满足了监控测量实时性的需求。

1 实时性能分析

1.1 嵌入式实时操作系统和μC/OS-II

嵌入式操作系统 (Embedded Operating System, EOS)主要负责嵌入式系统的全部软、硬件资源的分配、调度、控制、协调并发活动。它必须体现其所在系统的特征，能够通过装卸某些模块来达到系统所要求的功能。

μC/OS-II 是专门为计算机的嵌入式应用而设计的实时操作系统，是基于静态优先级的占先式（preemptive）多任务实时内核。采用 μC/OS-II 作为测试的目标，一方面是因为它已经通过了很多严格的测试，被确认是一个安全的、高效的实时操作系统；另一个重要的原因，是因为它免费提供了内核的源代码，通过修改相关的源代码，就可以比较容易地构造自己所需要的测试环境，实现自己需要的功能。

1.2 实时操作系统和系统实时性能指标

实时系统对逻辑和时序的要求非常严格，如果逻辑和时序出现偏差将会引起严重后果。实时系统有两种类型：软实时系统和硬实时系统。软实时系统仅要求事件响应是实时的，并不要求限定某一任务必须在多长时间内完成；而在硬实时系统中，不仅要求任务响应要实时，而且要求在规定的时间内完成事件的处理。通常，大多数实时系统是二者的结合。

事实上，没有一个绝对的数字可以说明什么是硬实时，什么是软实时。它们之间的界限是十分模糊的。这与选择什么样的 CPU，它的主频、内存等参数有一定的关系。另外，因为应用的场合对系统实时性能要求的不同而有不同的定义。因此，在现有的固定的软、硬件平台上，如何测试并找出决定系统实时性能的关键参数，并给出优化的措施和试验数据，就成为一个具有普遍意义并且值得深入探讨的课题。

因为采用实时操作系统的意义就在于能够及时处理各种突发的事件，即处理各种中断，所以衡量嵌入式实时操作系统的最主要、最具有代表性的性能指标参数无疑应该是中断响应时间了。中断响应时间通常被定义为：

中断响应时间=中断延迟时间+保存 CPU 状态的时间+该内核的 ISR 进入函数的执行时间。

中断延迟时间=MAX(关中断的最长时间，最长指令时间) + 开始执行 ISR 的第一条指令的时间。

通俗点定义就是：从中断发生起，到执行中断处理程序的第一条指令所用的时间。由于实时操作系统更多考虑的是最坏的情况，而不是平均的情况，因此指令执行的时间就按照最长的指令执行时间来计算，所以中断延迟时间，通常是由关中断的最长时间来决定的。

2 总线式数据采集系统的组成与功能

该系统采用总线巡检方式，对监测对象进行数据采集与处理，系统硬件以模块化结构，实现 32/64/128 路模拟或数字量的集中监测，适用于各种标准现场一次仪表或二次仪表数据测量与控制。整机采用先进的微机处理技术和通信控制技术，并嵌入实时处理内核，智能化程序较高，工作性能更加

基金项目：重庆市科委自然科学基金资助项目(CSTC 2005BB2220)
作者简介：金 纯(1967－)，男，博士、研究员；吴小波，硕士生；王时龙、陈 峰，教授、博士
收稿日期：2006-04-03 **E-mail：**wxb@jinoux.com

稳定，测量精度高，通用性强。

2.1 系统组成

该系统的硬件组成如图 1 所示。

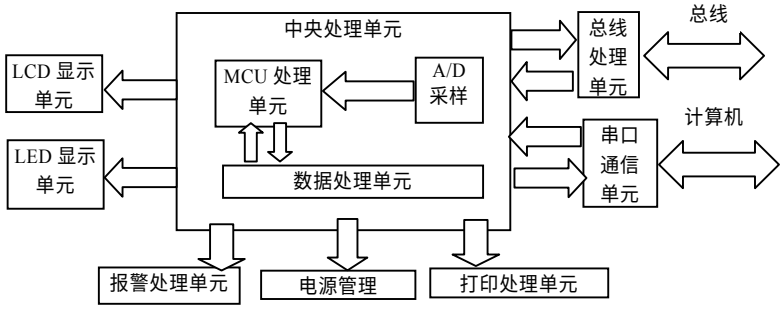


图 1 总线式数据采集系统硬件组成方框图

现场监测通道状态以总线方式，通过总线处理单元传送到中央控制单元进行数据采集与处理，其中 CMU 采用具有 10 位 A/D 转换器的 80C196KB。

2.2 系统功能

该系统可以对各通道的工作参数、状态进行即时修改设定，并可以通过面板 LED 实时显示 32/64/128 路通道的工作状态，同时各通道的实时参数通过 LCD 进行逐屏显示。对发生报警的信道可以通过打印处理单元进行打印输出、声光报警及显示。该系统采用总线巡检方式，对各信道工作状态进行远程数据采集并进行集中数据处理。为进一步满足智能化管理的需要，具有和计算通信的功能，可以实现监测数据的共享。同时，也可以通过计算机对各信道的工作状态进行设置，进一步增强了该系统的智能化管理能力。

3 μC/OS-II 在系统中的应用

该系统若采用以往的前后台式工作方式，即后台为主应用程序，前台为中断处理程序；通常情况下执行主程序，若有中断发生时，则转向前台处理中断服务程序。前台需要处理定时显示系统信息子程序，或按键中断处理子程序，然后根据中断程序中所置的状态标志，由主程序判断其状态标志后再进入相应的子程序，也就是主程序采用状态查询方式进行工作。这样在一定程序上不能保证整个系统测量的实时性。因为主程序在执行其它程序时，不可能随时去检测这些状态标志，尤其是处理多信道 A/D 采样计算时，耗时较多。当工作的信道增加或减少时，这种现象则表现得尤为明显，而且难以实现并行操作的相互通信。在主程序的各个子模块中，有需要横向联系交换信息的，这在一般的前后台系统中是很困难的，且存在系统不稳定的隐患。实时内核兼具实时多任务性和稳定性，因此考虑采用实时内核。MC/OS-II 是一个源码开放的实时内核，且又有许多成功的先例可供参考，可针对不同的 MCU/MPU，通过条件编译裁减其内核的大小，以满足系统要求。MC/OS-II 是占先式内核，总是运行就绪条件下优先级最高的任务。最大可以管理 64 个任务，其中集留 8 个给系统，故应用程序最多可以有 56 个任务。鉴于许多成功先例和系统成本，采用了 80C196KB 作为系统的 MCU。通过实验，基本满足了系统所要求的实时性。

3.1 开发实时内核的流程

开发实时内核的流程如图 2 所示。

3.2 内核的移植

内核的移植也就是使实时内核能够在某个微处理器或微控制器上正常运行。移植工作包括以下几个内容：

(1)在 OS_CPU.H 中用#define 定义 3 个宏，声明 C96 中能够识别的数据类型和堆栈的增长方向。

(2)在 OS_CPU.C 中用的 C 语言重新编写以下几个函数：OSTaskStkInit,OSStartHighRdy,OSTaskCresteHook,OSTaskSwHook,OSTaskDelHook,OSTaskStatHook,OSTimeTickHook。

(3)在 OS_CPU.ASM 中编写几个汇编语言函数 LoadCtx()、OSCtxSw()、OSIntCtxSw()、OSTickISR()。

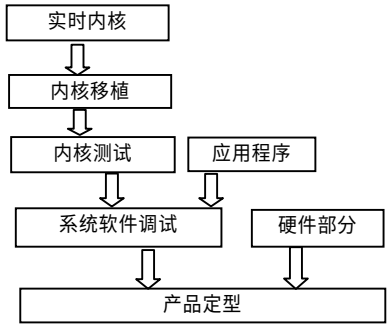


图 2 实时内核开发流程

3.3 实时内核在应用中应注意的问题

一个实时系统的软件由实时操作系统加上应用程序构成。应用程序与操作系统的接口通过系统调用来实现。用 80C196KB 作为系统的 MCU，只能用内部 RAM 作为 TCB 和所有系统存储器（含各种控制表）以及各个任务的工作和数据单元。因此一定要注意以下几点：

(1)为各个任务分配各自的堆栈区，该堆栈区既作为任务的工作单元，也作为任务控制块的保护单元。

(2)系统的任务控制块只存放各任务的堆栈指针，而任务的状态均存放于任务堆栈中。在一个任务退出运行时，通过中断把它的状态进栈，然后把它的堆栈指针保存于系统的 TCB 中；再根据优先取出优先级最高的已就绪任务的堆栈指针 SP 映像值送入 SP 中；最后执行中断返回指令转去执行新任务。

(3)各任务的数据和工作单元尽量用堆栈实现，这样可以允许各任务使用同一个子程序。使用堆栈实现参数传递并作为工作单元，而不使用绝对地址的 RAM，可实现可重入子程序。该子程序既可为各个任务所调用，也可实现递归调用。

3.4 应用 μC/OS-II 实时内核的主要部分

3.4.1 任务的分配

实时系统中的任务有别于前后台系统中的子程序模块，任务是处理机按程序处理数据的过程，是个动态的概念。一般一个任务对应于一段独立的主程序，它可能调用各种子程序，并使用各种系统资源如中断、外设等，以完成某种选定的功能，且允许多个任务并行。根据该系统的性能指标和技术要求，可对系统进行如下的任务划分：按键中断，LCD 显示，串行通信，打印与报警，信道巡检 A/D 采样与数据处理，系统信息显示，系统工作参数测量，电源切换与充电管理共 8 个任务。

3.4.2 任务的调度

μC/OS-II 的任务调度是按优先级进行的，根据各任务的实时性要求及重要程度，分别置它们优先级为 4、9、8、7、6、11、10、5。其中 0、1、2、3、OS_LOWEST_PRIO-3、OS_LOWEST_PRIO-2、OS_LOWEST_PRIO-1、OS_LOWEST_PRIO 这几个优先级保留以被系统使用。优先级号越低，任务的优先级越高。这样程序之间的通信可以通过按键中断置标志来实现，其中按键中断的优先级最高。当其它任务运行时，按键中断将使得系统服务转向运动按键中断处理子程序 ISR。当中断处理子程序运行完后，转向判断就绪状态任务的优先级别。如果发现有比中断前任务优先级更高的任务，则

转向执行该任务。先判断其运行标志，如果是“非”，则又等待。再重复上述过程。如果在执行完 ISR 后发现没有比中断前任务优先级更高的，则转向中断前的子程序继续运行。该系统的软件处理没有采用优先级转换的方法，而是采用状态置位判断的方法，这样可以减少程序的复杂性。

3.4.3 任务间的通信

任务间通信最简便的方法是使用共享数据结构。虽然共享数据区法简化了任务间的信息交换，但是必须保证每个任务处理共享数据时的排它性，以避免竞争和数据的破坏。通常与共享资源打交道时，使之满足互斥条件最一般的方法有：关中断，使用测试并置位，禁止任务切换，利用信号量。

在本系统中采用了前两种。关中断是一种最简单快捷的方式，也是在中断服务子程序中处理共享数据结构的唯一方法。要注意的是：关中断的时间尽量短，以免影响操作系统的中断处理。其应用模式如下：

```
void Function(void)
{
OS_ENTER_CRITICAL()；
…… /*在此处理共享数据*/
OS_EXIT_CRITICAL（）；
}
```

测试并置位方式需要有一个全局变量，约定好先测试该变量；如果是约定的数值，则执行该任务，否则不执行该任务。这种方法称测试并置位（TEST-AND-SET）或 TAS。其应用程序如下：

```
Disable interrupts /*关中断*/
If (Access Variable is 0){
/*若资源不可用，标志为 0*/
Set variable to 1; /*置资源不可用，标志为 1*/
Reenable interrupts； /*重开中断*/
Access the resource； /*处理该资源*/
Disable interrupts; /*关中断*/
Set the Access variable back to 0; /*清资源不可使用，标志为 0*/
Reenable interrupts； /*重新开中断*/
}else{ /*否则*/
Reenable interrupts； /*开中断*/
*/资源不可使用，以后再试*/
}
```

3.4.4 时钟节拍

时钟节拍是特定的周期性中断，根据本系统的性能指标，取 1ms。时钟的节拍式中断使得内核可以将任务延时若干个整数时钟节拍，以及当任务等待事件发生时，提供等待超过的依据。另外，系统信息的定时显示需要系统每隔一次的时钟节拍显示一次。

3.4.5 存储空间的分配

为了减少操作系统的体积，只应用操作系统的任务调度、任务切换、信号量处理、延时及超时服务几部分。这样可使该操作系统的大小减小到 3KB～5KB，再加上应用程序最大可达 50KB 左右。因为每个任务都是独立运行的，所以每个任务都具有自己的栈空间。这样可以根据任务本身的需求（局部变量、函数调用、中断嵌套等）来分配其 RAM 空间。

4 结束语

总之，μC/OSII 实时操作系统开发应用程序有其独到之处，用户可以直接利用系统的接口函数编写自己的应用程序，不需另行开发，大大方便了用户编程，缩短了软件的开发周期，提高了开发效率。随着各种应用电子系统的复杂化和系统实时性需求的提高，并伴随应用软件朝着系统化方向发展的加速，μC/OS-II 实时内核一定会得到更大的发展。因为它可以使产品更加稳定可靠，开发过程更加规范。

参考文献

1 Labrosse J J. MicroCOS-II: The Real Time Kernel[M]. America: CMP, 2001.

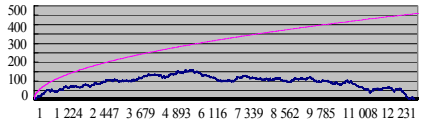
2 Labrosse J J. μC/OS-Ⅱ：The Real Time Kernel[M]. America: R&D Books, 2002.

3 Labrosse J J. 嵌入式实时操作系统 μC/OS-II[M]. 邵贝贝, 译. 北京：北京航空航天大学出版社, 2003.

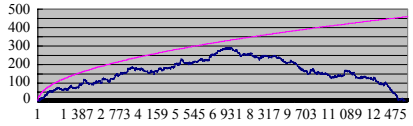
4 Labrosses J J. 嵌入式系统构件[M]. 袁勤勇, 译. 北京 :机械工业出版社, 2002.

(上接第 258 页)

(2)索引表中点数的验证



(a)距离最短路径



(b)时间最短路径

图 6 索引表中结点数的变化

选择最大跨度的两点进行距离最短路径和时间最短路径的计算，得到索引表中点数的变化曲线如图 6 所示。图 6 中的坐标横轴为 P 结点的个数，也就是计算的进度，纵轴是索引表中的结点数，进行距离最短路径的计算时，索引表中出现的最大结点个数为 163。进行时间最短路径计算时，索引表中出现的最大结点个数为 294。图中的平滑曲线为 $4\sqrt{P}$ 。

参考文献

1 严蔚敏, 吴伟民. 数据结构(C 语言版)[M]. 北京: 清华大学出版社,1997.

2 王开义, 赵春江, 胥桂仙, 等. GIS 领域最短路径搜索算法的高效实现[J]. 中国图像图形学报, 2003, 8A(8): 951-956.