基于 uC/OS-II 的设备驱动框架设计

李辉武，唐　平，陈天葆
（广东工业大学自动化学院，广东 广州 510006）

【摘　要】针对 uC/OS-II 没有提供一个完整的设备驱动框架，文章根据 uC/OS-II 自身的结构特点，借鉴其中任务间通信手段，设计了一个设备驱动框架，通过该驱动框架为设备建立驱动程序并规范相应的 API 函数。最后探讨了 uC/OS-II 下设备驱动程序具体开发流程。

【关键词】嵌入式实时操作系统；uC/OS-II；设备驱动；API 函数
【中图分类号】TP311　　【文献标识码】A　　【文章编号】1008-1151(2008)11-0067-02

（一）前言

uC/OS-II 是 JEAN J.LABROSSE 编写的一个源码公开、可移植、可固化、可裁剪、抢占式的嵌入式实时操作系统。uC/OS-II 能管理 64 个任务，并提供任务调度与管理、内存管理、任务间同步与通信、时间管理和中断服务等功能，具有执行效率高、占用空间小、实时性能优良和可扩展性强等特点。但是与其他嵌入式内核相比， uC/OS-II 没有设备管理功能。虽然 JEAN J.LABROSSE 针对嵌入式系统常见设备驱动另外编写了一本著作 Embedded Systems Building Blocks(《嵌入式系统构件》)，但还是没有为 uC/OS-II 构造一个完整的设备驱动体系。

在使用 uC/OS-II 进行驱动开发时，由于没有规范的 API 函数，驱动程序的可移植性和可读性明显下降，这影响了 uC/OS-II 的推广和应用，所以规划一个简单、稳定、规范的设备驱动体系结构非常必要。文献[3]中已经针对 uC/OS-II 设计了一个通用的驱动框架，实现对系统设备的有效管理。但在该驱动框架下，设备驱动程序和 uC/OS-II 内核是分别独立的，在小型应用系统中，程序太大，不易于裁剪。本文根据 uC/OS-II 自身的结构特点，设计了一个设备驱动框架，将设备驱动程序设计为内核模块。这样在实际应用中，通过对操作系统配置文件进行设置，就能得到一个既满足需要，又非常紧凑的应用软件系统。

（二）设备驱动概述

设备管理即输入输出子系统，可分为上下两部分：一部分是上层的，与设备无关，这部分根据输入输出请求，通过特定的设备驱动程序接口来与设备进行通信。另一部分是下层的，与设备有关，常称为设备驱动程序，它直接与相应设备打交道，并且向上层提供一组访问接口。

设备管理的目标是对所有外接设备进行良好的读、写、控制等操作。由于用户希望能用同样的应用程序和命令来访问设备和普通文件。为此，LINUX 中的设备管理应用了设备文件这个概念来统一设备的访问接口，对设备进行操作的系统调用和对文件的类似，主要包括 open()、read()、write()、ioctl()、close()等。

嵌入式系统将设备分成3类：字符设备、块设备及网络设备。其中字符设备和块设备与虚拟文件系统(VFS)相关，并为操作系统提供了统一的操作接口。字符设备和块设备的主要区别是前者在读写设备时不需要缓冲区，而后者需要。内核与网络设备驱动间的通信完全不同于与字符设备以及块设备驱动程序之间的通信，而是采用了一套和数据报传输相关的函数，如BSD的套接字(socket)接口。

（三）uC/OS-II 的设备驱动框架

在《嵌入式系统构件》中，JEAN J.LABROSSE 根据嵌入式系统的硬件结构框图，将和设备操作相关的软件模块封装为不同的“任务”，例如“键盘任务”、“显示任务”、“通信任务”等，然后通过信号量来实现任务之间的“行为同步”。考虑到 uC/OS-II 的结构特点，为了增强内核的灵活性和为了方便，本文设计了设备驱动框架模型，如图 1 所示：

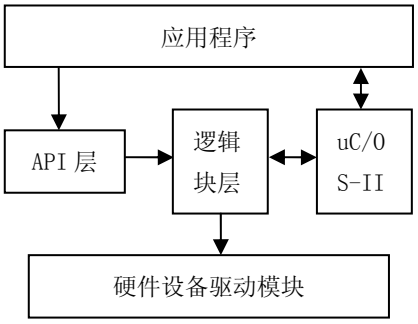

图 1　设备驱动框架模型

在这个设备驱动框架中，API 层只为应用程序提供了 3 个 API 函数:DeviceOpen()、DeviceClose()、CreateDevice()；而逻辑块层实现了对核心数据结构和通用设备的操作函数的封装。下面是核心数据结构和相关函数：

1.借鉴 uC/OS-II 中的管理事件控制块方法，定义一个设备控制块的数据结构。设备控制块的定义如下：

```
typedef struct device_tcb {
```

【收稿日期】2008-07-28

【作者简介】李辉武（1981－），男，广东工业大学自动化学院硕士研究生，研究方向为智能控制系统及其应用；唐平，女，广东工业大学自动化学院教授，研究方向为计算智能与智能机器人进化算法。

```
INT8U device_id;          //设备 ID
INT8U device_type;       //设备类型
OS_EVENT * device_sem;  //设备信号量
struct device_tcb * OSDevPtr;   //下一个设备控制块指针 DEV_OPERATIONS * dev_op_handle;   //设备操作句柄}DEV_TCB
```

在 uC/OS-II 初始化时，在初始化函数 OSInit()中按应用程序使用设备总数 OS_MAX_DEVICE,创建 OS_MAX_DEVICE 个设备控制块，并借用成员 OSDevPtr 作为链接指针，把这些空设备控制块链成一个单向链表。

2.设备的操作句柄函数列表包括：设备安装、读出、写入；写入和读出其实质是写和读各个设备的缓冲区；而控制应包括设备参数的设置、起动和停止三部分。设备打开和关闭主要解决设备的共享和分配。

```
Typedef struct device_operations_list{
//设备控制列表
 INT8U (*DevWrite)(INT8U *str,INT32U length);
//设备字节写函数
 INT8U (*DevRead)(INT8U *str,INT32U length);
//设备字节读函数
 INT8U (*DevControl)(void *param);
     //设备控制函数
 OS_EVENT* (*DevInstall)(void *param,INT8U *err); //设备安装与设备控制函数的差别在于设备安装要传递向量相关信息
 DEV_TCB * parent;
} DEV_OPERATIONS;
```

3.设备打开的过程是从有效的设备控制块链表中找到指定 ID 的设备控制块，审请信号理，如果审请通过则返回该设备的操作句柄。

```
DEV_FUN * DeviceOpen(INT8U DeviceID,INT8U * err){
     DEV_TCB  *pevent;
     INT8U OSerr;
     pevent = OSDevFreeList;
while(pevent->OSDevPtr != (DEV_TCB *)0 ){
     if(pevent->device_id == DeviceID) {
              //找到指定 ID 的设备
 OSMutexPend(pevent->device_sem,0,&OSerr);
 //向操作系统审请打开设备操作
     if(OSerr != OS_NO_ERR){
        *err = DEV_WAIT_TIME_OUT;
      return NULL ;
      }
     *err = DEV_NO_ERR;
return (DEV_FUN*)pevent->dev_op_handle;//利用强制转换继承函数接口并返回设备接口
     }
     pevent = pevent->OSDevPtr;
  }
     *err = DEV_NO_FIND_DEV;
     return NULL;
}
```

该函数以任务独占方式打开一个设备文件，当一个设备被占用时其它任务只有等待，直到占有设备的任务调用关闭设备文件函数释放该设备。

（四）设备驱动程序开发流程

1.对设备进行初始化。初始化的目的是使设备处于某种工作状态。以便应用程序访问该设备。串口初始化包括设置串口波特率、数据位、奇偶校验位、停止位等。以及把相关的驱动程序(读/写、中断服务函数等)，注册到设备控制列表中，以便系统对其管理及使用。

2.从设备上接收数据并提交给系统。这项功能通常就是所说的读操作，接收部传输来的数据。

3.把数据从主机上发送给设备。这项功能对应通常的写操作，把主机上的数据传送给外界。

4.对设备进行控制操作。在使用设备过程中，有时根据应用的需要对设备进行控制。例如，在使用串口过程中，有时需要改变串口波特率的设置。

4.编写设备驱动程序的注册函数。这个函数的功能主要是创建设备，从空闲设备控制块列表中取得一个控制块，同时调用操作系统的创建互斥信号量函数初始化 device_sem。初始化设备操作函数列表指针，最后将该控制块链接到有效设备控制块链表中。

5.编写设备驱动程序的卸载函数。当系统不需要使用某设备时，为了节省资源开销，需要把当初分配给被卸载设备的设备控制块从设备控制块链表中删除，并归还给空设备控制块链表。

（五）结束语

uC/OS-II 仅是一个实时内核，它只提供了任务管理、任务的通信同步和简单的存储管理三项基本服务，因此在实际应用中需要对 uC/OS-II 进行功能扩展。本文为 uC/OS-II 设计的设备驱动框架，在为上层应用程序提供了良好的设备访问接口的基础上，充分利用 uC/OS-II 已有的功能。同时，该设备驱动框架没有破坏 uC/OS-II 易于移植、裁剪的特点，极大地方便了用户开发设备驱动程序。

【参考文献】

[1] JEAN J. LABROSSE. 嵌入式实时操作系统 uC/OS-II(第二版)[M].邵贝贝,译.北京:北京航空航天大学出版社.2003.

[2] JEAN J. LABROSSE. 嵌入式系统构件[M].袁勤勇,等译.北京:机械工业出版社,2002.

[3] 吴绍根. uC/OS-II 下通用驱动框架的设计与实现[J].微计算机信息,2006.

[4] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman. Linux 设备驱动程序(第三版)[M].魏永明,等译.北京:中国电力出版社,2006.

[5] 钟坚文,蔡旭.基于 uC/OS-II 的 CAN 总线驱动程序设计[J].微计算机信息,2005.