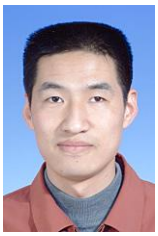
基于 I/O设备驱动机制的 CAN设备驱动程序设计

李 军

(中国航空计算技术研究所,陕西 西安 710068)



摘 要:设备驱动设计是嵌入式系统开发的难点。在分析 VxWorks实时操作系统字符型 I/O设备驱动机制的基础上,给出了 VxWorks下 CAN设备驱动设计的一般步骤和设计思路,为 VxWorks其他设备驱动开发提供了参考。

关键词:VxWorks操作系统; CAN总线; I/O系统;设备驱动

中图分类号: TP273 **文献标识码**: A **文章编号**: 1671-654X(2008)02-0082-03

引言

CAN总线作为工业现场总线的一种,因其具有较高的位速率和极高的抗电磁干扰能力,能侦测和处理产生的任何总线错误,并且具有高可靠性、实时性和灵活性,因此被广泛的应用到工业控制、汽车电子设备管理、航空、航天电子星和箭载等领域。在众多现场总线协议中,CAN总线协议由于在装备自动控制实现中是应用最成功的一套总线协议。随着信息技术的进步和快速发展,基于 CAN总线通讯的一些嵌入式系统越来越广泛地应用于工业设计中,VxWorks作为美国风河公司 (WindRiver)推出的具有工业领导地位的高性能商用实时操作系统 (Real - Time Operation System, RTOS)也越来越被广泛推广和应用。本文详细分析了 VxWorks设备驱动机制,并通过 CAN驱动设计,实例给出了 VxWorks设备驱动设计的一般思路。

1 VxWorks下字符设备驱动接口

对于嵌入式系统驱动程序的开发,主要解决的问题就是对硬件的控制和访问,实际上这种控制可以分为两种方式:一种是 CPU通过专用的驱动程序来实现对硬件设备的访问,这一类设备驱动按照程序员的编码习惯实现,且与硬件设备耦合太紧密,设备驱动的移植性较差;另一种就是通过操作系统提供的 I/O系统与设备驱动程序的挂接,用户通过标准的 I/O系统所提供的服务进行访问硬件设备,通常这类设备的驱动程序比较复杂,但移植性较好。因此,在实际应用中通过 I/O系统访问设备不但可以很好的解决其移植性,而且增强了设备驱动程序的可维护性。

1.1 设备驱动接口和组成

通常,在 VxWorks操作系统中,字符型设备的基本操作为接收和发送一个字符流,与 UNIX类似,VxWorks中提供了 open(), creat(), read(), write(), ioctl(), close()和 remove()等这些统一接口实现对设备的访问[1]。所有的设备都被视为文件,用户只需要定义文件名建立相应的文件描述符,随后的操作均由文件描述符用统一接口展开调用。

VxWorks中 I/O系统层次见图 1。

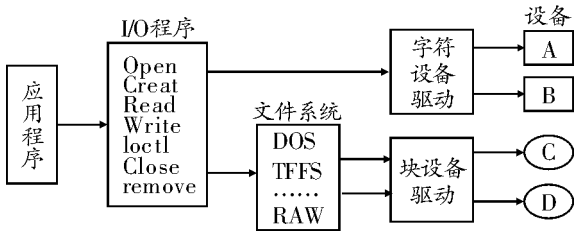


图 1 VxWorks操作系统 IO系统层次图

VxWorks中,应用程序通过文件名及文件描述符与硬件设备进行交互,一个完整的字符设备驱动包含设备驱动表、设备列表和文件描述符表。设备驱动表存储 I/O系统中设备驱动的入口,该驱动表的大小由宏 NUM_DRIVERS定义,系统中的设备驱动列表可通过 iosDrvShow命令查询显示。设备列表是一个双向链表,包含每个设备驱动的基本信息及其在设备驱动表中的设备号,设备列表支持多个设备访问相同的设备驱动,即设备驱动支持多设备。文件描述表中每一项

收稿日期: 2007-11-26 **修订日期**: 2008-02-20

作者简介:李 军 (1977 -),男,陕西彬县人,工程师,研究方向为计算机应用技术。

由设备驱动号和设备描述符 (该表中通常以设备 D标识)组成,本表的大小由宏 NUM_FILES决定。

1.2 设备驱动与 I/O系统挂接

根据 1.1节中的相关描述,设备驱动与 I/O系统主要通过设备驱动列表和设备表的相互作用来实现的。

第 1步,通过 iosDrvInstall (pCreate, pDelete, pOpen, pClose, pRead, pWrite, pIoctl)在驱动列表中安装用户的驱动,通常这 7个参数是可选的。create和 open的操作入口相同, delete操作入口为 NULL。若执行成功则返回该设备驱动在驱动列表中的驱动号。

第 2步,在成功安装好驱动后,将调用 iosDevAdd (pDevHdr, name, drvnum)添加由 name指定的设备。设备列表中提供对 open(), create()和 delete()三个函数从设备名称到设备驱动的匹配,成功添加设备后,在 Shell下通过 devs或 iosDevShow命令查询当前添加的设备驱动信息。

第 3步,执行 fd = open (devName, flags, 0x0)以文件的方式打开该设备,其中 flags是文件操作的读写模式标志。若该操作执行成功, I/O系统把驱动号和设备 D加入到文件描述符表中,此后, read, write和 ioctl函数通过 fd来访问设备。

2 CAN设备驱动开发

由于 CAN设备是字符型设备,应用程序可直接通过驱动程序访问 CAN设备。因此对于 CAN设备通讯的需求和 VxWorks操作系统 I/O系统的要求,基于 CAN设备的驱动开发主要是针对系统 iosInstall例程中的 7个入口函数的开发。从 CAN设备驱动的功能需求出发,可将 CAN设备驱动划分为 3个部分:初始化部分,函数功能部分和中断服务程序。

2.1 CAN设备描述定义

在 VxWorks操作系统中,每个设备都要有一个连接到系统设备链表上的设备描述结构体,该结构体必须包含一个 DEV_HDR类型的域, DEV_HDR独立于 CAN设备,另外还需要包含一些与 CAN设备相关的数据、信号量、中断信息及其他与设备密切相关的配置信息, DEV_HDR的作用是将 CAN设备描述结构体添加到系统设备描述的双向链表中[2]。

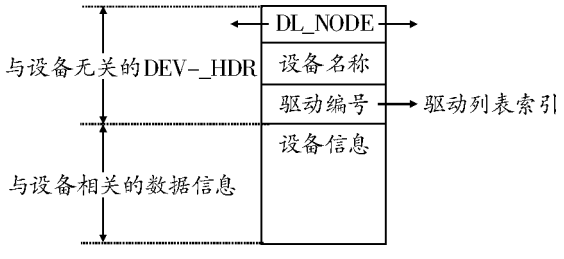


图 2　CAN设备描述结构示意图

CAN设备描述结构示意见图 2。

根据图 2示意, CAN设备名称可根据应用系统对 CAN的需求进行定义,本文中假定应用系统只存在一个 CAN设备,因此可命名为“/CAN0”,其中“CAN0”可表示第 0个 CAN设备,在此设备下可能存在多个 CAN通信通道,本文中假定只有一个 CAN通信通道,所以在“CAN0”设备下定义第 0个 CAN通信通道“/CAN0/1”,该标识在 VxWorks下被认为文件。

2.2 CAN设备描述生成

在 VxWoks的 I/O系统中,每个设备下是可以存在多个文件的,但 DEV_HDR只包含了设备头信息,与文件相关的信息需要保存在设备描述结构体里。为了能够生成 CAN设备 D,还需要定义单独 CAN通道设备信息 CAN_CHAN,该信息包含两个域:第一个域必须是指向 CAN_DEV设备描述结构体的指针 pDev,另外一个为 CAN通道信息位 chan。从 DEV_HDR到设备 D的转换是在标准 open操作中完成的。本文以 pCANOpen (DEV_HDR * pDevHdr, char * name, int flags)为例给出了一个 CAN设备描述生成的设计流程,并返回的 CAN设备 D,该流程图见图 3。

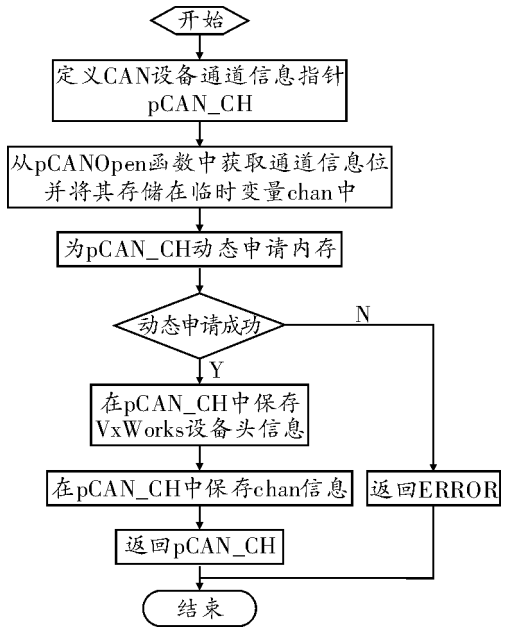


图 3　CAN设备描述生成的设计流程示意图

系统通过调用 fd = open (“/CAN0/1”, o_read, 0x0),在设备列表中查找相应的设备名“/CAN0”,并获取 DEV_HDR中的 CAN设备驱动索引号 drvNum。根据 drvNum系统在驱动列表中取得 pCAN_Open的执行入口地址并执行该函数,把其返回的 CAN设备 D

信息作为一条新的信息存储在文件列表中,将新信息索引号赋值给 fd[3]。

2.3 CAN设备驱动安装

定义 CAN 设备其他功能函数 pCANIoctl(devId, options), pCANRead(devId, pBuf, number)和 pCANWrite(devId, pBuf, number),驱动安装函数 drvCAN()和设备安装函数 devCANCreate()。

drvCAN()将 CAN 设备功能函数 pCANOpen、pCANRead和 pCANWrite添加到设备驱动列表中并返回设备驱动索引号 drvNum。devCANCreate()把设备"/CAN0 添加到系统设备双向链表中,同时完成 CAN 设备的初始化、中断配置和联接。根据 3.2节中所述,通过 open()打开设备文件"/CAN0/1 "所返回的 fd,系统的应用程序通过 read(fd, pBuf, numbe)和 write(fd, pBuf, numbe)函数对 CAN设备进行读写访问。

2.4 CAN设备数据读写方式

CAN设备在进行数据通讯时可通过查询方式和中断方式进行,在 CAN设备初始化后,通过 pCANIoctl(devId, options)对 CAN 工作方式进行设置,同时该函数可在 CAN设备在工作方式下对进行其他参数设置,如波特率配置、输出方式和报文验收滤波设置等。查询工作方式的编程较简单,而中断模式需要借助二进制信号量来实现[4]。中断方式下,系统初始化时通过调用 semGive()获取读、写信号量,以保证读、写 CAN设备任务首次被系统调度时因无法获取信号量而被系统挂起。

当 CAN设备接收到总线数据时产生接收中断,在中断服务程序中通过释放读信号量,以唤醒被操作系统挂起的读任务,读任务从 CAN设备的 FIFO中读取数据并将其写入 CAN设备读环形缓冲 mgCANRD中。同样当 CAN设备发送缓冲区空时产生中断,在中断服务程序中释放写 CAN设备发送数据信号量,以唤醒写 CAN设备任务从发送环形缓冲 mgCANWR中提取数据发送。

2.5 CAN设备数据对外接口

上述主要描述了基于操作系统 I/O 下的 CAN设备驱动程序,为了实现具体的应用,可以根据具体需求进行 CAN设备数据对外接口的设计,这个接口主要是实现对 CAN设备读写环形缓冲的访问。

3 结束语

本文将不同类型的 CAN控制器抽象为基于 VxWorks操作系统的 I/O系统上 CAN设备,提出了一种 CAN设备驱动设计方法,并在工程上得到了实现和验证,试验证明数据通讯及数据交换状况良好。基于 I/O系统的 CAN设备驱动所采用的开发方式集成和封装了 CAN控制器 I/O访问,使 CAN设备在 VxWorks的 I/O系统下被作为一种特殊文件,与其他设备组织成为一致的更高层次的抽象,通过 open(), read()和 write()函数为使用者和操作系统提供了统一的系统服务和用户接口。

参考文献:

[1] 陈新,唐震洲,胡倩. VxWorks操作系统下 D设备驱动的开发[J]. 大众科技, 2007, 93(7): 37-38.

[2] 解月江,张梅. VxWorks下设备驱动技术研究[J]. 航天控制, 2004, 22(6): 55-57.

[3] Wind River. VxWorks5.5 BSP Developer's Guide4[EB]. WindRiver system Inc, 1998.

[4] 谢伟,周忠丽,王忠仁. VxWorks下驱动程序的设计[J]. 中国民航飞行学院学报, 2005, 16(4): 40-41.

Design of CAN Device Driver Based on Mechanism I/O Device Driver

LI Jun

(Aeronautical Computing Technique Research Institute, Xi'an 710072, China)

Abstract: Design of device driver is the difficulty in embedded system's development. Based on the character I/O device driver mechanism analysis of real-time operation system (RTOS) of VxWorks, a general method and design idea of CAN device driver under VxWorks was presented in this paper, and it may be refered to the other device driver developing based on VxWorks OS.

Key words: VxWorks OS; CAN bus; I/O system; device driver