CN43-1258/ TP
ISSN 1007-130X

计算机工程与科学
COMPUTER ENGINEERING & SCIENCE

2009 年第 31 卷第 5 期
Vol. 31 ,No. 5 ,2009

文章编号:1007-130X(2009)05-0118-03

# 基于 MC9S12DP512 和 μC/ OS-II 的 CANopen 主站开发*

# Development of the CANopen Master Based on MC9S12DP512 and μC/ OS-II

徐 喆,闫士珍,宋 威,张 卓

XU Zhe,YAN Shi-zhen,SONG Wei,ZHANG Zhuo

(北京工业大学电子信息与控制工程学院,北京 100022)

(School of Electronic Information and Control Engineering,Beijing University of Technology,Beijing 100022,China)

**摘 要**:本文给出一种基于 MC9S12DP512 和 μC/ OS-II 的 CANopen 主站解决方案,主要阐述了 μC/ OS-II 的移植、MC9S12DP512 底层驱动模块的编写和 CANopen 主站的结构及测试结果。

**Abstract**:This paper provides a solution of developing the CANopen master based on MC9S12DP512 and μC/ OS-II,and illustrates the translation of μC/ OS-II, the development of the MC9S12DP512 driver, and the analysis of the CANopen master structure and the test results.

**关键词**:CANopen;μC/ OS-II;驱动;调度机

**Key words**:CANopen;μC/ OS-II;driver;scheduling machine

**中图分类号**:TP336

**文献标识码**:A

## 1 引言

CANopen 应用层协议在国内外已经有很多方面的应用,受到了广泛的重视。在欧洲,CANopen 协议已被广泛应用于医疗装置中,并进一步扩展应用到保安控制系统中;在美国,CANopen 协议已经成为装载机械和公共运输设备的协议标准,同时也应用于嵌入式控制系统。本文实现的 CANopen 主站系统是基于 μC/ OS-II 嵌入式实时操作系统结构的,该 CANopen 主站核心是基于调度机的结构,具有可扩展性和实时性,可与汽车各电子控制单元构成车载网络或用于大型设备、工业现场等。

## 2 CANopen 简介

CANopen 是基于 CAN 总线协议的应用层协议。CANopen 预研阶段是 1995 年在以 Bosch 公司为首的团队带领下开发的,其说明文档递交到 CiA(CAN in Automation,简称 CiA)国际用户及生产厂商组织,现在已经成为国际标准 CENELEC EN50325-4。

对象字典是 CANopen 的核心,它描述了设备使用的所有数据类型、通信参数和应用参数。CANopen 设备之间通过交换通信对象进行通信。为了能够实现通信、网络管理和紧急情况处理等功能,CANopen 协议定义了四类标准的通信对象:用于实时数据传输的进程数据对象(PDO);在设备之间传输数据,用来配置 CANopen 网络上设备的服务数据对象(SDO);监控从设备节点状态的网络管理报文(NMT);CANopen 还定义了三个特殊功能对象如同步对象、时间标签对象和紧急对象[1]。

## 3 硬件平台介绍[2]

本系统使用飞思卡尔 MC9S12D 系列单片机 MC9S12DP512。该系列采用 5V 供电,总线速度为

* 收稿日期:2008-01-04;修订日期:2008-04-02
基金项目:北京市科技新星基金资助项目(2003A005);北京市教委重点项目和北京市自然科学基金共同资助项目(KZ20041000501)
作者简介:徐喆(1968-),女,广东五华人,博士,副教授,研究方向为网络控制系统、推理控制和软测量。
通讯地址:100022 北京市朝阳区平乐园 100 号北京工业大学电子信息与控制工程学院;Tel:13683510522;E-mail:xuzhe@bjut.edu.cn
Address:School of Electronic Information and Control Engineering,Beijing University of Technology,100 Pingleyuan Rd,Chaoyang District,Beijing 100022,P. R. China

25MHz;主要用于工业控制,特别适合用在汽车上;具有丰富的 I/O 模块和工业控制专用的通信模块。其 80 引脚封装的单片机有 59 个 I/O 端,112 引脚的有 91 个 I/O 端。通信模块有 SCI、SPI、I²C、CAN、J1850、Bit Flight 等,片内(如 MC9S12DP512)最多集成有五个 CAN 模块。另外,MC9S12DP512 片内有 4K EEPROM、14K RAM、512K Flash EEPROM,无需进行片外程序存储器的扩展。

# 4 μC/OS-II 的移植与测试

μC/OS-II 是一种面向中小型系统的高效、实时、可扩展的嵌入式操作系统,其内核提供任务调度与管理、时间管理、任务间同步与通信、内存管理和中断服务等功能,最小可编译至 2KB,可以在绝大多数 8 位、16 位和 32 位的处理器、微控制器和 DSP 上运行。

操作系统的移植就是使一个实时内核能在某个微处理器或微控制器上运行。为了方便移植,大部分的 μC/OS-II 代码是用 C 语言写的,但仍需要用 C 和汇编写一些与处理器相关的代码。要使 μC/OS-II 正常运行,处理器必须满足以下要求[3]:

(1)处理器的 C 编译器能产生可重入代码;

(2)用 C 语言就可以打开和关闭中断;

(3)处理器支持中断,并能产生定时中断(如 10～100Hz);

(4)处理器支持的硬件堆栈(如几千字节);

(5)处理器有将堆栈指针和其它 CPU 寄存器读出并存储到堆栈或内存中的指令。

MC9S12DP512 能满足以上要求,所以 μC/OS-II 能在该芯片上运行。

图 1 给出了 μC/OS-II 与硬件系统的关系[3]。其中,INCLUDES.H 是主头文件,在所有.C 文件的开始都包含了该文件。若使用 INCLUDES.H,所有.C 文件都只包含一个头文件,程序简洁,可读性强。该文件的程序清单如下:

```
# INCLUDE  "MC9S12DP512.H"
# INCLUDE  "μCOS_II.H"
# INCLUDE  "OS_CPU.H"
# INCLUDE  "OS_CFG.H"
# INCLUDE  "Main.H"
# INCLUDE  "Datatype.H"
```

其中,Datatype.H 是定义 CANopen 主站代码中使用的数据类型的头文件。


图 1 μC/OS-II 的文件结构图

OS_CPU.H 包含了与处理器相关的数据类型、宏和结构体,因此 μC/OS-II 的移植需要重新定义这些数据结构;OS_CPU_C.C 中必须实现 OSTaskStkInit()函数,用来完成对任务堆栈的初始化操作;OS_CPU_A.ASM 中必须实现下列四个函数:

(1)OSStartHighRdy():用来运行优先级最高的就绪任务;

(2)OSCtxSw():任务级的任务切换函数;

(3)OSIntCtxSw():中断级的任务切换函数;

(4)OSTickISR():时钟节拍产生的中断服务函数,本应用中时间间隔定义为 20.48ms。

测试移植 μC/OS-II 的正确性并不复杂:首先建立一些简单的线程和时钟节拍中断服务程序,若各线程能被正常调度,就可以添加应用程序到线程了。本系统测试时建立了三个线程:AppStartTask、AppTask1 和 AppTask2。AppStartTask 线程创建线程 AppTask1 和 AppTask2 之后将自己挂起。线程 AppTask1 等待信号量 EventSem1 并发送信号量 EventSem2;线程 AppTask2 等待信号量 EventSem2 并发送信号量 EventSem1。不同的线程中包含不同的串口输出语句,操作系统开始启动线程调度后,首先可以看到 AppStartTask 线程中的输出语句,之后线程 AppTask1 和 AppTask2 中的串口输出语句交替输出。这说明,AppStartTask 线程之运行一次便将自己挂起不再运行,线程 AppTask1 和 AppTask2 交替运行。测试证明,在 MC9S12DP512 上移植的 μC/OS-II 内核运行正常。

# 5 驱动模块的开发

本主节点的外设资源包括 SCI、CAN 和定时器,由相应寄存器组控制运行。为这些模块编写一组驱动程序,以供应用程序利用。

## 5.1 串行通信接口(SCI)驱动模块

串口驱动主要有 SCI 初始化子模块、单个字符输出子模块、字符串输出子模块和整数(包括 8 位、16 位和 32 位)输出子模块。其中,SCI 初始化子模块用于配置波特率和使能 SCI;整数输出模块是将传递给该驱动模块的整数原样输出。

## 5.2 CAN 驱动模块

CAN 驱动主要包括 CAN 控制器初始化子模块、CAN 报文发送子模块、CAN 报文接收子模块、中断配置子模块。CAN 控制器初始化子模块用于 CAN 总线波特率和滤波器的配置以及 CAN 控制器时钟的选择。CAN 消息接收模块将报文从 CAN 控制器中读到报文结构体 CANMsg 中。CANMsg 结构体定义如下:

```
typedef struct _CANMsg {
    char   rtr;
    char   msglen;
    int    cobid;
    char   data[8];
    struct _CANMsg  * pPre;
    struct _CANMsg  * pNext;
} CANMsg , *pCANMsg;
```

其中,rtr 是 CAN 报文远程帧标识,msgLen 是报文中的数

据长度,cobid 是报文的 ID,data[8]存储报文中的数据。该驱动的对外接口是一个指向 CANMsg 的指针。CAN 消息发送模块将报文从 CANMsg 读出并写进 CAN 控制器的相应寄存器中,其对外接口也是指向 CANMsg 的指针。中断配置子模块用于配置 CAN 控制器的中断。

### 5.3 定时器驱动模块

定时器驱动包括定时器初始化子模块、系统当前时间获取子模块、时间比较子模块和时间差值子模块。定时器初始化子模块负责时钟的配置和定时器的启动;系统当前时间获取子模块通过定时获取当前微秒级时间提供给 CANopen 使用。时间比较和时间差值子模块分别用于两个微秒级时间大小比较和求取时间差。

以上各模块驱动通过调试运行正常。

## 6 CANopen 主站系统的开发与测试

### 6.1 基于 MC9S12DP512 和 μC/OS-II 的 CANopen 主站软件系统结构

如图 2 所示,开始线程、调度机线程、写报文线程是用户建立并由 μC/OS-II 内核管理的三个线程。开始线程是操作系统完成初始化工作之后开始调度线程时调度的第一个线程。它主要完成硬件的初始化,包括串口和系统时钟的初始化、时钟节拍的初始化、CAN 的初始化、中断的初始化以及创建调度机线程、写报文线程,该线程运行一次之后就将自己销毁;调度机线程主要完成 CANopen 内部各任务的调度,在没有更高级线程处于就绪态或者出现中断时该线程将会一直处于循环运行状态。

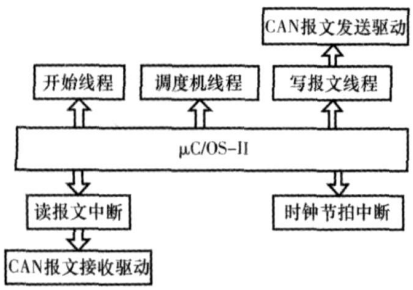


图 2 CANopen 主站系统结构图

写报文线程负责将需要发送的报文写进 CAN 控制器相应的寄存器中,并将其发送到 CAN 总线上,该线程由调度机线程在需要的时候启动;当 CAN 控制器从总线上收到报文时,将产生接收中断,该中断将触发读报文中断服务程序执行。读报文中断服务程序通过调用 CAN 报文接收驱动程序,从 CAN 控制器中读出收到的报文。

### 6.2 任务调度机

根据主站的实时性和可扩展性的要求,采用多任务调度机[4],其具体结构如图 3 所示。它由任务等待队列和任务运行队列两种基本队列组成,任务运行队列包括九个优先级的子队列;执行选择、执行任务、销毁任务、运行选择由调度机主函数完成,构成一个循环体;而其他的各部分都由任务组成。

对于单个任务而言,其运行流程如图 4 所示。

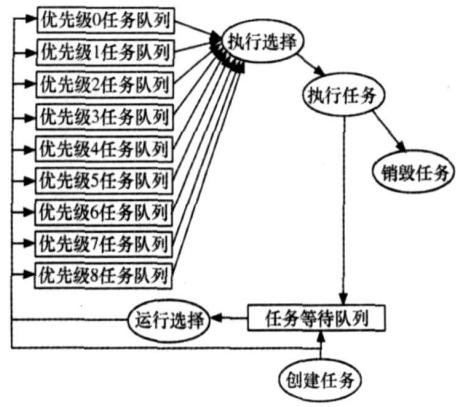


图 3 调度机结构图

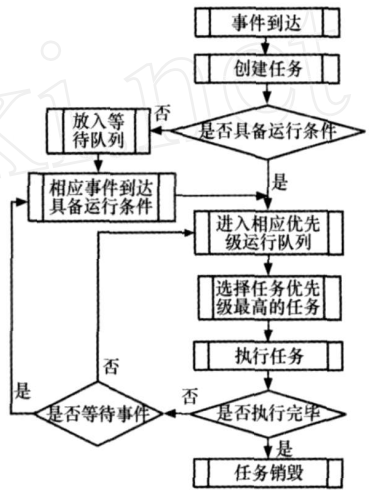


图 4 任务运行流程图

一旦任务获得执行,它的运行、停止、终结都由任务自身决定。由于在纯 C 环境中很难做到可剥夺的任务调度,因而任务函数自身应可以做到分段执行、自行阻塞。当任务返回时,调度机函数依据返回信息对其做进一步处理。本系统规定任务有以下返回值:

(1) TASK_OK:任务运行完毕,可以销毁;

(2) TASK_WAIT:任务自身阻塞,放入等待队列;

(3) TASK_RUN:任务可继续运行,放入执行队列;

(4) TASK_ERR:任务运行出错,进行异常处理。

该调度机利用两种基本队列(任务等待队列和任务运行队列)管理 CAN 报文接收、CAN 报文发送、PDO 报文分发、SDO 报文分发、NMT 报文分发、同步报文生成、PDO 处理、SDO 处理和 NMT 处理等任务。这些任务生成之后被放入任务等待队列,交由调度机管理。当有任务满足运行条件时,被调度机移入对应优先级的任务运行队列中。调度机还负责从运行队列中获取最高优先级的任务执行,并根据任务返回值决定对任务下一步的处理。调度机内部任务的关系如图 5 所示。

有报文到达时,读报文中断服务程序运行,在该中断服务程序中触发 CAN 报文接收任务。CAN 报文接收任务开始运行,将接收到的报文放入 PDO、SDO、NMT 报文接收队列,并触发相应的报文分发任务,以触发对应的处理任务。如果在 PDO 处理、SDO 处理和 NMT 处理任务中有报

(下转第 125 页)

drivers: A New Architecture for Device Drivers[C] //Proc of the 11th Workshop on Hot Topics in Operating Systems, 2007:85-90.

[13] Fraser K, Hand S, Neugebauer R, et al. Reconstructing I/O[EB/OL]. [2007-11-23]. http://www.ccci.com/tools/ttcp/.

[14] Härtig H, Hohmuth M, Liedtke J, et al. The Performance of μ-Kernel Based Systems[EB/OL]. [2007-11-21]. http://os.inf.tu-dresden.de/pubs/sosp97.

[15] Vasseur J L, Uhlig V, Stoess J, et al. Unmodified Device Driver Reuse and Improved System Dependability via Virtual Machines[C] //Proc of the 6th Symp on Operating Systems Design and Implementation, 2004:17-30.

[16] Forin A, Golub D, Bershad B. An I/O System for Mach 3.0[C] //Proc of USENIX Mach Symp, 1991:163-176.

[17] Liedtke J. On μ-Kernel Construction[C] //Proc of the 15th ACM Symp on Operating System Principles, 1995:1-14.

[18] Maren K T V. The Fluke Device Driver Framework:[Master's Thesis][D]. University of Utah, 1999.

[19] Bershad B N, Savage S, Pardyak P, et al. Extensibility, Safety and Performance in the SPIN Operating System[C] //Proc of SOSP '95, 1995:267-284.

[20] Engler D R, Kaashoek M F, Jr J O. Exokernel: An Operating System Architecture for Application-Level Resource Management[C] //Proc of SOSP '95, 1995:251-266.

[21] Hunt G. Creating User-Mode Device Drivers with a Proxy [C] //Proc of the 1997 USENIX Windows NT Workshop, 1997:123-131.

[22] Satyanarayanan M, Mashburn H H, Kumar P, et al. Light Weight Recoverable Virtual Memory[C] //Proc of SOSP '93, 1993:146-160.

[23] Seltzer M I, Endo Y, Small C, et al. Dealing with Disaster: Surviving Misbehaved Kernel Extensions[C] //Proc of OSDI '96, 1996:213-227.

[24] Wahbe R, Lucco S, Anderson T E, et al. Efficient Software-Based Fault Isolation[C] //Proc of SOSP '93, 1993:203-216.

(上接第 120 页)

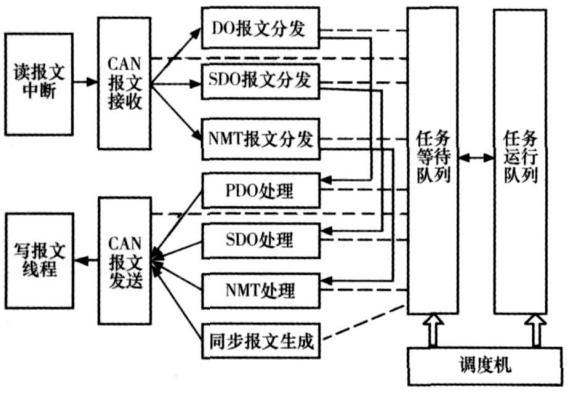


图 5 调度机内部任务之间关系结构图

文要发送或同步报文生成任务运行,将触发 CAN 报文发送任务。CAN 报文发送任务运行时将启动写报文线程。

### 6.3 内存管理

本主站的任务和报文管理采用动态链表。对象字典用散列表实现,即我们使用一个下标范围比较大的数组来存储对象,设计一个哈希函数使得每一个对象的索引和子索引都与一个函数值(数组下标)相对应,然后用这个数组单元来存储这个对象。但是,不能够保证每一个对象的索引和子索引与函数值是一一对应的,因此极有可能对于不同的对象,却计算出了相同的函数值,这样就产生了冲突现象。本系统解决冲突采用链地址法,这里同样要用到动态链表。动态链表的结构是动态地分配内存,即在需要时才开辟一个节点的存储单元。这种结构简单灵活,节省内存空间,便于查找、添加和删除节点。

### 6.4 主站通信功能测试

所实现主站与 BECKHOFF 标准的 CANopen 设备 BK5120 耦合器,组成包含两个 CANopen 节点的网络进行测试。系统测试如图 6 所示。

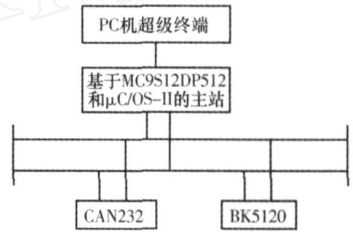


图 6 系统测试框图

图 6 中,本文开发的系统作为主节点,BK5120 耦合器作为从节点,CAN232 转换器用于监控 CAN 总线上的数据,PC 机的超级终端用于监控主节点的运行情况。经过测试证明,基于 MC9S12DP512 和 μC/OS-II 的 CANopen 主节点能完成从节点配置、网络管理、PDO 报文的收发和同步报文的发送等功能。本系统运行正常、稳定。

## 7 结束语

本文介绍了基于 MC9S12DP512 &μC/OS-II 平台的 CANopen 主站的开发方法,所实现的主站和标准的 BECKHOFF CANopen 设备通信。测试结果证明,该主站运行正常、稳定。希望本设计对嵌入式应用及 CANopen 领域的发展能够起到抛砖引玉的作用。

### 参考文献:

[1] CiA. CiA Draft Standard 301: CANopen Application Layer and Communication Profile Version 4.02[S]. 2002.

[2] Motorola Inc. MC9S12DP512 Device Guide V01.25[Z]. 2005.

[3] Labrosse J J. 嵌入式实时操作系统 μC/OS-II[M]. 第 2 版. 邵贝贝译. 北京:北京航空航天大学出版社,2003.

[4] Song Wei, Yan Shizhen, Xu Zhe, et al. Transplantable CANopen Master Based on Non-Preemptive Task Scheduler [C] //Proc of the IEEE Int'l Conf on Automation and Logistics, 2007:557-562.