[自控·检测]

DOI:10.3969/j.issn.1005-2895.2014.01.013

基于CANopen的套色系统主站的实现

徐 闯,邓忠华

(华中科技大学 自动化学院,湖北 武汉 430074)

摘 要:为了解决数据传输速度等问题,我们将CAN总线和CANopen协议应用到套色系统中。文中给出了一种基于CANopen协议和DOS操作系统并在工控机上实现套色系统监控主站的方法。基于实时性和可移植性的考虑,提出了一种用C语言实现的非抢占式的多任务调度结构。测试结果表明:所设计的CANopen监控主站运行稳定可靠,能很好地实现通信功能。

关 键 词:套色系统;CANopen协议;主站;任务调度;图形用户界面

中图分类号:TP311.1 **文献标志码**:A **文章编号**:1005-2895(2014)01-0054-04

Implementation of Color Register Control System Master Based on CANopen

XU Chuang, DENG Zhonghua

(School of Automation, Huazhong University of Science & Technology, Wuhan 430074, China)

Abstract: To solve the problem of data transmission, CAN bus and CANopen protocol were introduced in color register control system. A method was represented to realize the monitor master of color register control system based on CANopen protocol and DOS. In consideration of real-time and transportability, an ANSI C based no-preemptive multi-task scheduling structure was proposed. The experiment results indicate that the CANopen monitor master runs stably and reliably and has good communication performance.

Key words: color register control system; CANopen; master; task scheduling; Graphical User Interface

CANopen协议是基于CAN总线的应用层协议。在开放的现场总线标准中,是最著名和成功的一种,已经在欧洲和美国获得广泛的认可。CANopen协议精炼透明,具有实时性和可靠性高、数据传输速率高和组网成本低等特点,而且支持不同的CAN设备间的互操作性,互换性,具有标准化、统一的系统通信模式和设备描述方式[1]。根据CAN和CANopen的特点及在工业控制中的优势,我们构建了基于CAN总线的套色控制系统,设计了基于CANopen的具有高实时性的监控主站和从站,文中主要介绍主站的设计方法。

1 基于CAN和CANopen的套色系统的整体构成

整个系统是由n个TMS320LF240系列的DSP控制子系统与工控机构成的主从分布系统,通信采用CAN总线标准接口。工控机和DSP通过接口电路接入CAN总线,见图1。

每个DSP系统能够独立地完成相邻两色的套印,同时与上位机保持联络将数据、状态传送至上位机。工控机负责系统的监控工作,将系统的状态参数和数据实时显示给用户,并将用户输入的参数和命令传送给下位机。本系统开发了基于PCI总线的CAN接口卡:采用PLX公司的PCI9052作为PCI总线接口芯片,CAN控制器选用Philips公司的SJA1000,CAN收发器采用PCA82C250[2]。

2 监控主站的软件实现

2.1 CANopen主站软件设计的整体结构

监控主站要实现3个基本的功能模块:①CANopen协议栈的实现;②人机界面的实现、显示、修改参数以

收稿日期:2013-08-28;**修回日期**:2013-10-10

作者简介:徐闯(1986),男,河北保定人,硕士研究生,主要研究方向为现场总线技术和嵌入式应用。E-mail:515486918@qq.com

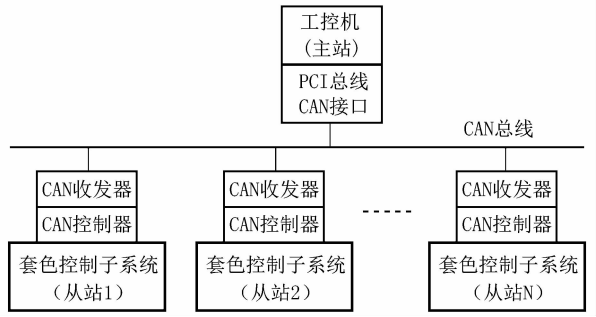

图 1　基于 CAN 总线的套色系统的整体结构

Figure 1　Structure of color register control system based on CAN bus

及传送命令;③消息队列,调度机的实现。整体结构如图 2 所示。

当 CAN 控制器接收到完整报文以后,触发 5 号中断,进入中断处理程序,根据 COB-ID 将报文分别放入 PDO,SDO,节点保护队列,当队列非空时,根据节点状态或其他触发条件,新建一个消息(Message),表征有接收报文需要处理并将新建的 Message 放入等待队列或者运行队列。对于人机监控程序,通过不断扫描鼠标和键盘的状态,触发相应的 Message 并放入运行队列或等待队列,调度机根据一定的算法,选择消息队列中的一个,调用其对应的处理函数进行处理。

2.2　CANopen 协议栈的实现

1) 对象字典。考虑到对象字典条目不是很多,所以采用了数组的方式构建对象字典,定义数组为 OD_ENTRY ObjDict[ ],OD_ENTRY 数据结构如下[3]。

```
typedef struct _vOD_ENTRY {
    BYTE   * pDATA   ;  //指向数据存储地址
    WORD   Index     ;  //索引值
    BYTE   SubIndex  ;  //子索引值
    BYTE   DatLength ;  //数据长度
    BYTE   EntryAttr ;  //属性:可读/写,只
                        读/写,可映射,有/
                        无符号
} OD_ENTRY;
```

对象字典的 2000h-5FFFh 项为制造商自定义规范区域,把从 2010h 到 201Ch 设为各通道的参数区,指向各个通道的套色参数[4]。

2) SDO 的传输方式有分段传输、加速传输和分块传输 3 种。根据实际需求,在系统运行过程中,需要更改参数或传送命令时,使用加速传输。在系统初始化时需要读取所有通道的全部参数,为了提高效率,使用分块传输。

3) PDO 的传输效率最高。把一些如印刷速度、误差等动态数据通过 PDO 从从站周期性地发给主站。

4) NMT 管理各节点的状态完成启动、复位、停止等功能。使用 HeartBeat 协议进行节点的监控和维护[5]。

2.3　消息机制及调度机的实现

处理接收报文、发送报文、处理鼠标操作、键盘操作等,这些都可以称之为 1 个任务(TASK)。完成 1 个 TASK 需要调用相应的函数。由于 CANopen 主站的特点,可能面临同时处理多个 TASK 的情况,所以既要满足实时响应又必须合理安排各个 TASK 执行的顺序。为此建立了 2 个消息队列:等待队列和运行队列。每当有 1 个 TASK 需要执行的时候,先建立 1 个 Message 放入队列中(满足执行条件的放入运行队列,需要等待某些触发条件的放入等待队列)。然后由调度机根据一定的算法决定哪个 TASK 先执行。Message 的结构如下[6]。

```
typedef struct _MessageObj{
    char      TaskType;       //任务类型
    char  *   toRunEvent;     //触发条件
    char  *   ( *pFun)(struct _MessageObj  *);
                              //任务函数
    char      TaskState;      //任务状态
    int       data[10];       //数据空间
    struct _MessageObj  *  pNext;  //下一个
    struct _MessageObj  *  pPre;   //前一个
} MessageObj;
```

调度机制:根据消息结构体中的任务类型(TaskType),将不同的任务划分为不同的优先级[7],定义见表 1。

表 1　任务优先级

Table 1　Task priority

级别	任务类型	级别	任务类型
0	接收中断	4	心跳报文处理
1	发送报文(SDO,NMT,SYNC 等)	5	状态机维护
2	PDO 报文处理	6	鼠标、键盘操作处理
3	SDO 报文处理	7	人机界面刷新

等待队列是直接由消息结构体组成的双链表,存储未满足触发条件的阻塞任务。运行队列比较复杂,见图 3。它由 8 条单独的队列构成,每个队列代表一个优先级。每条队列有 1 个标志位,代表队列空/非空。按照优先级从高到低的顺序检查标志位,当标志位为 1 时,就可找到优先级最高的待执行任务。调度机流程见图 4。

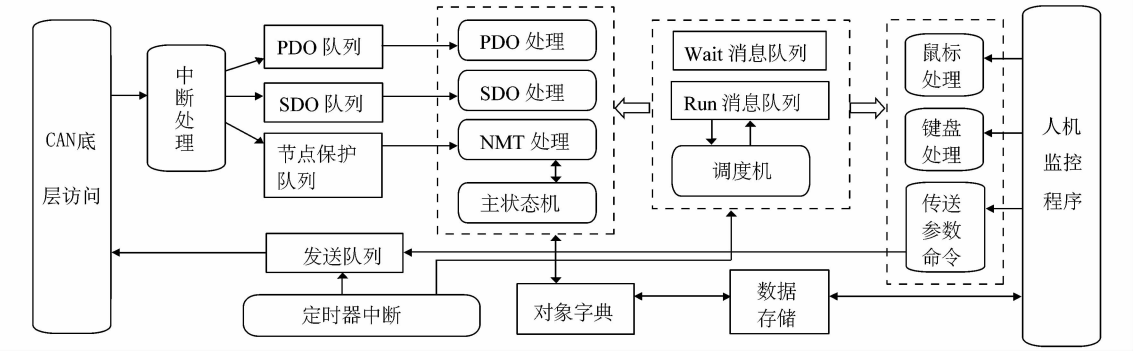

图 2 主站软件的整体结构

Figure 2 Software structure of the master

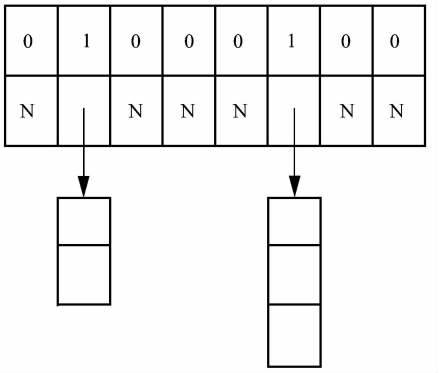

图 3 运行队列

Figure 3 Run queue

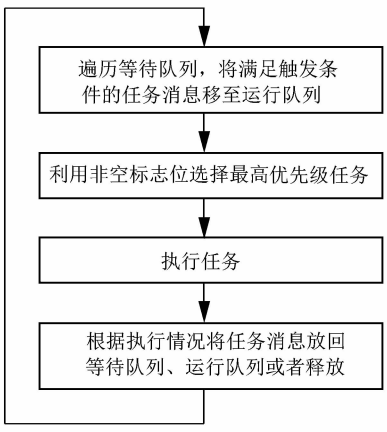

图 4 任务调度执行流程图

Figure 4 Flowchart of task scheduling

2.4 人机界面的设计

1）为了满足人机界面数值显示，数值输入，历史曲线，换页等不同功能，用 C 语言开发了不同的功能元件（component），每一个元件根据其功能有一个元件类型（COMPONTYPE），不同的元件类型对应不同的处理函数。定义如下：

```
typedef enum _COMPONTYPE
{
    PAGEBUTN        =0 , //换页按钮
    STATEBUTN       =1 , //复状态按钮
    DATADISP        =2 , //数值显示框
    ADDBUTN         =3 , //增,减值按钮
    ERROCURVE       =4 , //误差历史曲线
    SCREENKEYBD     =5 , //屏幕键盘元件
    ……
    FAKECMP         =31 //空元件
}COMPONTYPE;
```

2）对于所有的元件，它们有一些公共的属性，记录在结构体 COMPONENT 中，定义数组 COMPONENT index_table[ ]，index_table[ ]中包含所有元件。同时每一种不同类型的元件为了实现特定功能，有自己独特的属性，为此对于每一个元件需要 2 个结构体进行描述。以换页按钮为例[8]，定义如下：

```
typedef struct
{
    COMPONTYPE type;//元件类型
    int x1;//元件位置
    int y1;//元件位置
    int width;//元件宽度
    int height;//元件高度
    int cmpattr;//元件属性
    ……
}COMPONENT;//公共属性
typedef struct
{
    char pagenum;//画面编号
    char butcolor[2];//按钮颜色(凹,凸)
    char * label[2];//按钮文字(中,英)
```

```
    char * imagename[2];//按钮图片(凹,凸)
}PAGEBUT;//换页按钮属性
```

3) 鼠标处理函数。当左击鼠标时,产生一个消息,消息类型是 MousLeftPress。并将鼠标位置 mousex, mousey 存在该消息的数据域。执行该任务时,代码如下:

```
for(i = page_comp[pagenum].comp[0];i <= page_comp[pagenum].comp[1];i++)//当前页搜索
{
    if((mousex > index_table[i].x1)&&(mousex < index_table[i].x1 + index_table[i].width) &&
    (mousey > index_table[i].y1)&&(mousey < index_table[i].y1 + index_table[i].height)&&
    ((index_table[i]. cmpattr &( Not_Used |LOCKED) == =0))//元件有效
    {
        CLP[index_table[i].type](i);//由函数指针数组调用相应的处理函数
    }
}
```

page_comp[pagenum].comp[0]记录当前页(页码为 pagenum)中起始元件的编号即在数组 index_table[]中的位置。page_comp[pagenum].comp[1]记录最后元件的编号。当鼠标位置在某一元件内部且该元件有效时,由 CLP 函数指针数组进入相应处理函数。CLP 定义如下:

```
void ( *CLP[])(int compnum)
{
    PagebtLp,     //换页按钮左击处理函数
    StatebtLp,    //复状态按钮左击处理函数
    DatdspLp,     //数值显示框左击处理函数
    AddbtLp,      //增减值按钮左击处理函数
    ……
};
```

3　测试与分析

将所设计的套色系统监控主站应用于武汉某工业自动化公司的实验平台,CAN 总线上连接 12 个通道即套色控制从站。依次检查各个通道、主站动态数据的显示、主站向从站传送参数、命令等功能均能很好地实现,同时满足实时性要求。工控机界面如图 5。

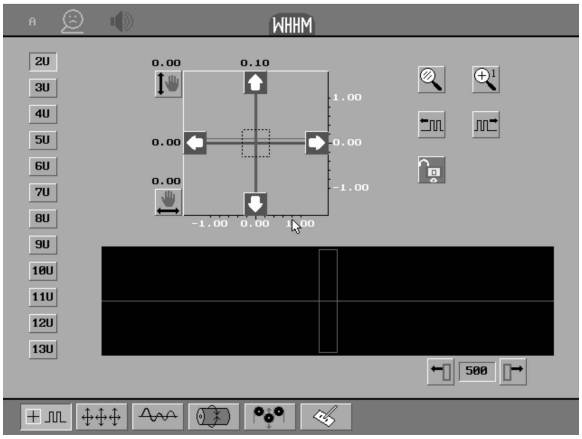

图 5　套色系统监控界面

Figure 5　Monitoring interface of color register control system

4　结语

文中提出了一种 C 语言实现的 CANopen 监控主站的实现方法。为了满足多任务处理和实时性的要求采用了一种模拟的消息机制和任务调度的方法。该方案结构简单,清晰具有一定的参考价值。当然还有很多需要改进的地方,比如调度机算法的改进[9],考虑各任务执行的时间,以及在一定时间内将未执行完的任务放回等待队列等待下次调度。进一步研究在 DOS 下提高屏幕分辨率的方法,以及随着图片的大量应用如何合理分配数据存储方式等。

参考文献:

[1] 饶怡欣,胥布工,匡付华. 基于 CANopen 的电动执行机构远程监控主站的实现[J]. 计算机测量与控制,2010,18(2):373-374.

[2] 鲍旭日,刘淑敏. 面向 CAN 总线接口卡的 PCI 接入技术[J]. 中国仪器仪表,2008(11):49-50.

[3] 徐喆,闫土珍,宋威. 基于散列表的 CANopen 对象字典的设计[J]. 计算机工程,2009,35(8):44-45.

[4] CAN in Automation (CiA). CANopen application layer and communication profile: DS-301[S]. Nuremberg:CiA,2002:79-82.

[5] CAN in Automation(CiA). CANopen framework for programmable CANopen devices: DSP-302[S]. 3rd ed. Nuremberg:CiA,2002:16-24.

[6] 宋威,方穗明,张明杰. 任务调度在 CANopen 主站设计中的应用[J]. 计算机测量与控制,2008,16(4):559-560.

[7] 张梅. 基于 DOS 的实时多任务系统的实现[J]. 测控技术,2002,21(6):41-42.

[8] 王金连. 用 C 语言在 DOS 下实现 WINDOWS 的只能按钮感知技术[J]. 电脑编程技巧与维护,1998(4):12-13.

[9] 李毅,贺洪江,袁左英. DOS 下实时多任务的设计方法[J]. 工矿自动化,2004,4(2):45-46.