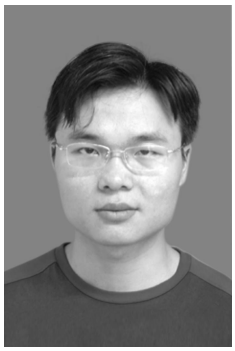
智能安防系统中单片机程序实现多任务机制方法的应用



王辉堂，　颜自勇，　陈文芗
(厦门大学 机电工程系，福建 厦门　361005)

摘　要：通过一个智能安防系统的应用实例，介绍了一种采用 MCS－51 单片机程序实现多任务机制的简单方法，并给出了源代码。采用中断进行实时任务切换，具有结构简单清晰、代码量少、不需使用汇编语言等优点。该方法亦可应用于其他单片机系统。

关键词：多任务系统；单片机；中断；安防系统
中图分类号：TP311.1　文献标识码：B　文章编号：1001-5531(2007)04-0048-50

王辉堂（1981—），男，硕士研究生，主要从事应用电子技术研究与嵌入系统设计。

Application of Multi-Task System Based on MCS－51 in Intelligent Security System

WANG Huitang,　YAN Ziyong,　CHEN Wenxiang
(Department of Mechanical & Electrical Engineering, Xiamen University, Xiamen 361005, China)

Abstract: A multi-task system based on MCS－51 was introduced, and an example for practical application of intelligent security system was presented. The real time task exchange is based on interrupt. The system is compact and convenient for application, which is programed without assemble language. The method is also available for the other MCU system.

Key words: multi-task system; MCU; interrupt; security system

0　引　言

传统的单片机程序一般采用单任务机制。单任务系统具有简单、直观、易于控制的优点。然而，由于程序只能按顺序依次执行，缺乏灵活性，在较复杂的应用中使用中断函数实时地处理一些较短的任务，极为不便。嵌入式多任务操作系统的出现解决了该问题。在多任务系统中可以有多个任务，各个任务是并行的，任务之间可以相互跳转。但是，嵌入式多任务操作系统在提供强大功能的同时，也带来了代码量大、结构复杂、对硬件要求较高、开发难度大、成本高等问题。

实时操作系统（Real Time Operation System, RTOS)的核心是中断，利用中断进行任务切换。在大部分 RTOS(如 μC/OS－II)中，每个任务都有自己的堆栈，用来保存任务的一些信息，任务之间通过信号量、邮箱、消息队列等传递信息。在很多情况下，只需要单片机在接收到控制信号后，切换到不同的工作状态，即只要进行任务切换，不需要保存任务的相关信息。舍弃这些复杂的功能，可以使程序结构变得简洁、易用。

本文设计的简单的多任务机制操作系统在只增加极少量 C 语言代码的前提下，无须使用汇编语言，也无须对原来的程序进行大改动，就可以实现多任务操作。

1　两种机制的比较

一个智能安防系统必须具备的功能有：① 当有人入侵时执行报警工作；② 用户可以通过键盘板进行功能设置；③ 主板能与管理中心进行通信；④ 当发生火灾、地震等灾情时，管理中心能及时通知用户。智能安全防系统结构如图 1所示。

颜自勇（1982—），男，硕士研究生，主要从事单片机嵌入式系统及现场总线技术的研究
陈文芗（1955—），男，教授，主要从事电子技术应用、计算机控制技术等的研究与教学工作。

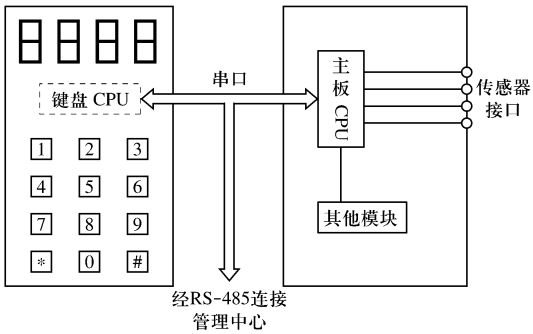


图 1　智能安防系统结构示意图

通常状态下,主板 CPU 不断地扫描各个传感器的状态。当检测到传感器的异常信号(如有人闯入)时, CPU 进入入侵报警状态,进行响警铃、拨打户主电话、通知管理中心等工作。用户可以从键盘板输入密码后解除报警状态。当发生火灾或地震时,管理中心发送一个串口代码给主板 CPU,使 CPU 进入灾难报警状态,进行响警铃、语音报警等操作。用户平时也可以通过键盘板使主板 CPU 进入功能设置状态。主板的程序如果采用单任务机制,其程序流程如图 2所示。

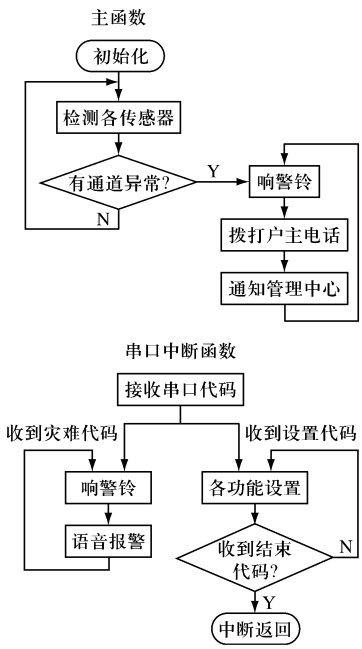


图 2　单任务机制程序流程图

在主函数中循环检测传感器状态,若有异常,则调用报警函数。灾难报警和功能设置由串口中断来完成。这种单任务结构有 2个缺点:① 在报警状态和功能设置状态下,程序需要不停地检测是否收到撤除信号。该要求在程序代码量大、执行工作较多的情况下很难实现。② 各状态之间的切换十分困难。用 C语言编写的程序为求模块化,一般函数数量较多,函数调用的嵌套层数也多。如果不想使用汇编语言,要从一个较深的嵌套立刻跳转到主函数是非常困难的。虽然使用库函数 setjmp()和 longjmp()可以实现长跳转,但这2个函数在中断函数内部是无能为力的。

2　实现多任务机制的程序结构

如果采用多任务机制的结构,则可以很好地解决上述单任务机制存在的 2个问题,如图 3所示。

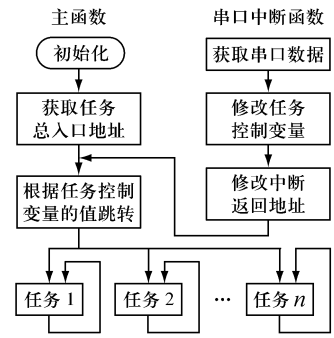


图 3　多任务结构

实现该多任务机制完整的源代码如下:

```
word idata PC_Value, SP_Value;     /*储存中断返回点、SP初值的全局变量
byte idata Ctrl_Code;     /*控制任务切换的全局变量,在中断函数里被赋值
void main()
{
Initial();     /*初始化函数,与程序结构无关
    SP_Value= SP;     /*获取 SP的初始值
    PC_Value=Get_Next_PC();     /*取下一条指令地址
    EA = 1;     /*获取 PC、SP初值后再开中断保证稳定性
    if(Ctrl_Code!  = 0)
    SP= SP_Value;     /*重置堆栈指针,防止堆栈溢出
    switch(Ctrl_Code)     /*任务入口地址,即中断的返回点
    {
        case 1:    goto    TASK1;
        case 2     goto    TASK2;
        case 3     goto    TASK3;
        default    break;
    }
TASK1:  for(;;)
```

```
    {
    //任务 0代码
    }
TASK2:  for(;;)
    {
    //任务 1代码
    }
TASK3:  for(;;)
    {
    //任务 2代码
    }
}
word Get_Next_PC(void)    //获取下一条指令的地址
{
    word address;
    address=*((unsigned char*)SP);    //PC的高字节
    address<<8;
    address+=*((unsigned char*)(SP-1));
    //PC的低字节
    return address+4;    //查看反汇编代码,计算所得
}
void Chuan_Kou_Interrupt(void) interrupt 4 using 0
{
    byte a1,a2,a1=a1*a2;
    *((unsigned char*)(SP-5))=PC_Value>>8;
    *((unsigned char*)(SP-6))=PC_Value&0x00ff;
    {
    //接收串口代码并根据代码修改 Ctrl_Code的值
    //其他操作
    }
}
```

3 任务调度原理与实现

程序的整体思路是利用中断进行任务切换。实现该目的需要解决3个关键问题:

(1) 获取任务入口点的地址。由于使用C语言不能直接获取和修改程序计数器PC的值,而在调用函数时会将PC值入栈,利用该特点在任务入口处之前调用Get_Next_PC函数即可从堆栈中获得入口地址。Get_Next_PC中,SP为堆栈指针,得到的PC值要加4才是任务入口地址,因为查看反汇编代码可知,将函数返回值传给全局变量PC_Value需要两条2 Byte的MOV指令。

(2) 修改中断返回地址。修改中断返回地址的操作与获取PC值类似,都是通过修改堆栈中的内容来实现的。但是由于编译器自身的特点,在进入中断时,编译器除了把返回地址入栈外,还会计算自身及它所调用的函数对寄存器ACC、B、DPH、DPL、PSW、R_0~R_7的改变,并将它认为被改变了的寄存器也入栈保护。这就给计算中断返回地址堆栈中的位置造成了困难。在中断函数定义时加上关键字using 0,通知编译器中断函数及其调用的函数将使用寄存器组0,这样工作寄存器R_0~R_7将不会被保存。ACC、PSW、DPH、DPL在对PC_Value操作时已经用到,在中断函数开头定义2个变量并令它们相乘,使B寄存器入栈。这样堆栈的结构就是固定的了。

(3) 防止堆栈溢出。在调用函数时编译器会将当前地址入栈,返回时再将其出栈。当任务切换即中断多次发生在函数调用过程中时,堆栈会因为只入不出而导致最终溢出,这是不能容许的。故在主函数初始化后应立刻保存SP值,在每次任务切换后都将SP恢复为初值,这样可以有效防止堆栈溢出。

在安防系统的例子中,可以把平时状态、入侵报警状态、危机报警状态和功能设置状态分别作为任务1、任务2、任务3和任务4,只要将对应的处理代码填入即可。CPU平时工作在平常状态,即任务1。检测到异常时,向键盘发送一串口数据并要求键盘回送。当串口收到数据时,根据代码的数值便可以迅速进行状态切换。

4 结　语

根据比较与分析可知,采用多任务机制的方法增加的代码量极小,无需使用汇编语言,其实时性好,移植简便,节省了程序开发的时间。该方法已经通过测试并应用于实际项目,取得了较好的效果。只要根据具体的硬件与编译环境稍作修改,亦可应用于其他的单片机系统中。

【参考文献】

[1] 张培仁. 基于C语言编程MCS-51单片机原理与应用[M]. 北京:清华大学出版社, 2003

[2] 胡大可,李培泓,张路平. 基于单片机8051的嵌入式开发指南[M]. 北京:电子工业出版社, 2003

收稿日期:2006-04-01