

现代计算机

18, 95.7

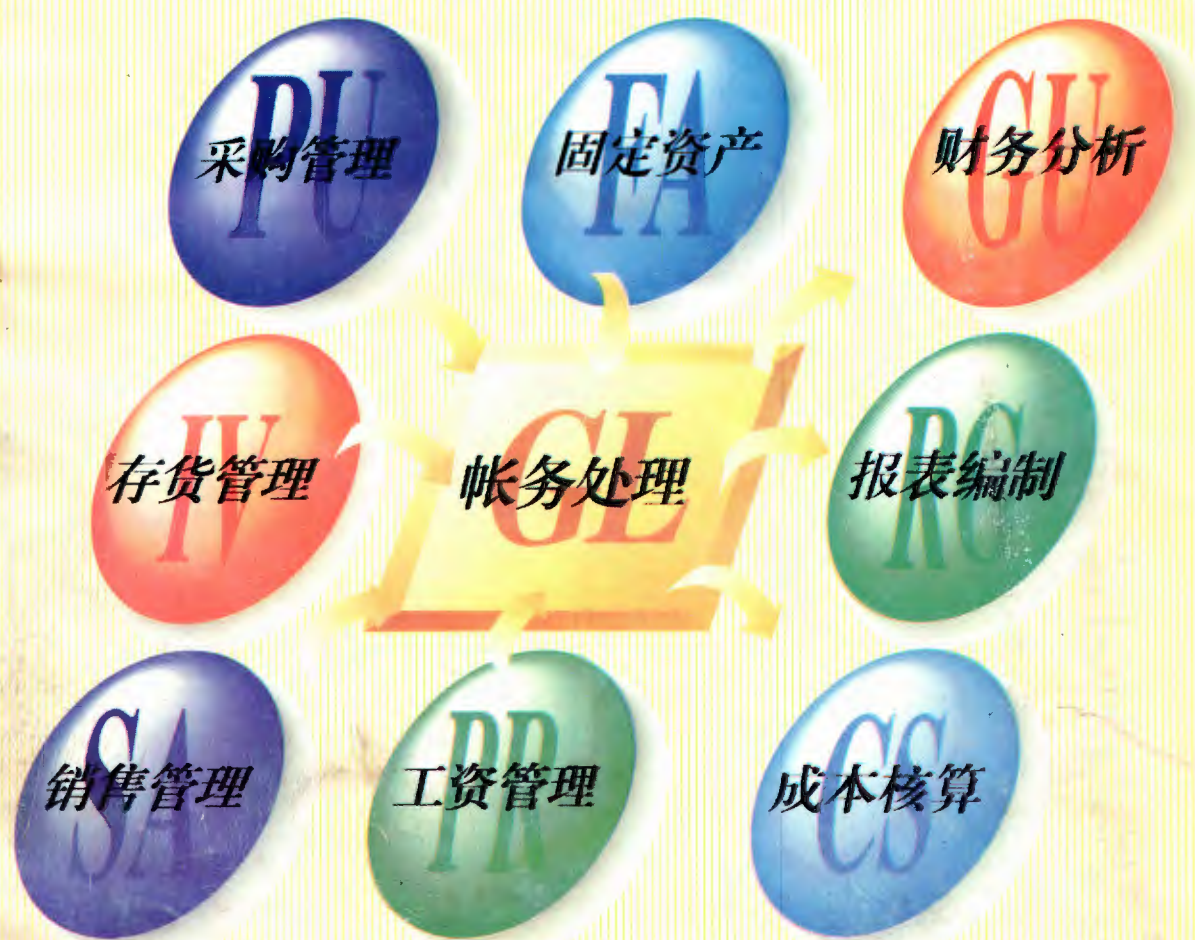
原名《电脑与微电子技术》

总第 42 期

MODERN COMPUTER

EASY 拓展财务网络系统

人算不如电算 理财首选拓展



广东劳业电脑系统开发公司

公司地址：广州市越秀南路 181 号 2 楼 邮编：510100

电话：3800115 3847499 3817370 - 205 206 207 传真：(020)3851054

40M² LED 全彩色电视广告屏 矗立在中国出口商品交易会门前

广州中鸣公司隆重推出



發展計算機
推動信息化

為現代計算機
題於九五年

七月

盧瑞華



盧瑞華副省長為本刊題詞

中山大学 电脑照排科技中心

博采众长 自成一格
技术先进 服务周到

经营： 高档精密照排系统
普及型轻印刷排版系统
电脑多媒体网络系统
书报刊排版印刷

承接： 印刷厂技术更新改造
新老排版系统换代升级
信息管理系统软件开发
电脑排版技术培训

我们的客户遍布茂名、珠海、湛江、深圳、阳江、广州
和省内外，欢迎垂询，欢迎更多的朋友加入我们的行列

诚聘： 电脑技术人员 2 名。条件：男性，大专文化以上，
年龄 30 岁以下。有意者请寄来自述简历及半身脱
帽相片一张，听候约见。一经录用，待遇从优。

地址：广州中山大学东北区 368 号 邮编：510275

经理：梁 勇 电话：4186300 1999

公开发行 1995年7月20日出版

主 办 中山大学
编辑出版 《现代计算机》杂志社
主 编 张伟铮
联系地址 广州新港西路 135 号 中山大学
邮政编码 510275
电 话 4186300—6540
刊 号 CN44—1415/TP
邮发代号 46—121
广告证照 粤工商广字 01110 号
印 刷 中山大学印刷厂
总 发 行 广东省报刊发行局
订 阅 处 全国各地邮局
每期订价 2.00 元 邮购价 3.00 元

本刊图文版权所有 未经允许不得转载

下 期 要 目

- 一种高精度温度测控系统的研制方法
- 灰色预测软件
- 新太网络应用平台
- 程控交换机与微机的串行通信
- 高级语言直接在打印机上绘图
- 单片机和单板机加密实用技术
- 非多媒体环境下 FLI 动画图像文件的应用
- 信息系统中库文件的设计原则与方法

目 录

☆ 研制·开发·应用 ☆

- 大型 LED 汉字智能显示屏研制····· 程启明 ②
灰色综合评估系统 ······ 朱诗兵 ⑦

☆ 网络与通信 ☆

- 计算机联网通信中调制解调器的研制
····· 肖南峰 林伟健 ⑩
网络管理信息系统共享数据的冲突处理
····· 梁京章 ⑭

☆ 编程技巧 ☆

- 通用汉化菜单编制技术 ······ 罗列异 ⑰
用图形成像技术生成各种新颖立体菜单
····· 晁永胜 ⑳
彩色立体窗口中文下拉式菜单编制技
巧和示例 ······ 黄焕如 ㉔
用 C 语言实现图形方式下的花样清屏
····· 丁永峰 ㉖

☆ 实用技术 ☆

- 如何在自制软件中加入五笔输入法 麦智祥 ㉚
屏幕图象彩色硬复制的 C 语言编程方法
····· 邓中明 刘光越 ㉜
通用嵌入式点阵汉字的实现与应用
····· 隋景明 孙迎春 ㉞
用五笔字型为 AutoCAD 输入汉字 兰文祥 ㉟

☆ 病毒与防治 ☆

- Gene 病毒的原理及其防治 ······ 卢胜文 ㉟
Greeting 病毒分析及清除 罗 明 朱德森 ㊲

☆ 软件纵横 ☆

- PC 机的键盘及其编程····· 王 革 ㊴
利用定时器中断预防键盘锁死 ······ 车光宏 ㊸

注重实用性 强调知识性
面向科研 面向生产 面向管理 面向用户 面向教学

大型LED汉字智能显示屏研制

江苏盐城工业专科学校 程启明

摘要:本文介绍 IBM-PC 和 8031 分布联合控制的大屏幕 LED 智能显示屏。该系统的显示屏幕由 128×256 个 LED 点阵构成,一屏最多可显示 8×16 个汉字,也可同时显示图案;IBM-PC 系统机主要完成各种显示文件的提取、编辑,并传输给单片机应用系统;8031 单片机应用系统完成对显示电路部分的控制,单片机先对显示电路的信息锁存器输入显示信息,然后控制扫描电路对显示屏 LED 点阵动态扫描,完成对文件的各种显示。IBM-PC 与 8031 之间采用并行通信进行数据传送。该显示系统结构合理、性能稳定可靠、显示方式丰富、显示内容修改方便,具有很好的实际应用价值,已在广告宣传等领域得到广泛应用。

关键词:智能显示屏 上、下位机 并行通信

一、引言

随着微电子技术的发展,显示器已成为人们直接获取各种信息的重要手段之一。目前显示器种类繁多,它们都由显示器控制电路和显示器件两部分组成。微机的出现及迅速普及应用,使显示器控制向智能化方向发展,真空型射线管等传统显示器件暴露出功耗高、体积大等缺点,已逐步被高质量的 LED 智能显示屏所取代。LED 智能显示屏的像素采用发光二极管,将许多发光二极管以点阵方式排列起来,便构成 LED 阵列显示屏幕。这种显示屏耗电少、成本低、清晰度高、寿命长,可用微机进行控制,因而操作控制灵活方便。它又有单色屏和彩色屏之分,有静态和动态两种扫描方式,也有独立固定和分布实时两种结构形式。独立固定显示系统一般采用单片机或单板机直接进行控制,需要显示的内容固化在 EPROM 存储器中,改变显示内容必须改写或更换 EPROM 芯片。分布实时显示系统采用系统机(即上位机)与单片机(即下位机)通信,通过系统上位机录入操作可随时向单片下位机传送数据,从而实现实时控制操作和信息显示。本文将介绍 IBM-PC 系统机和 8031 单片机主从实时控制的大屏幕 LED 智能显示屏。该显示屏可应用到工业、交通、商业广告、新闻发布、体育比赛、模拟军事演习、电子景观等广泛领域。

二、系统结构

本显示系统由 IBM-PC 系统上位机、8031 单片下位机、驱动电路、显示屏及电源等组成。其框图见图 1。

IBM-PC 上位机完成对图文数据的录入、编排和设置控制命令(如变化方式、动感、颜色等命令)以及对数据进行排序、缩放、平滑、转向等动态处理和按序向下位机显示缓冲区发送屏幕显示数据。8031 下位机按显示屏发光点(或发光模块)发送显示数据,解释变化方式、动感、颜色等控制命令并对显示屏发送控制信号。扫描电路完成在一定时间内对显示屏动态重复扫描。驱动电路则接受来自单片机的命令并把要显示的图文信息传到显示屏上显示。直流电源向各部分电路供电。各单元互相配合,各司其职。

显示缓冲区存放每屏显示数据及控制信号,其大小对应于显示屏上发光点的多少。单色屏上一个点对应于缓冲区中 1 个位,每个位的值(0,1)对应于显示屏上发光点状态(熄,亮)。彩色屏上一个点对应于缓冲区的几个位,其中第 1 个位的值对应于显示屏上发光点的状态,余下 $n-1$ 个位的值对应于显示屏上发光点的颜色, n 的大小决定颜色种类 M , $M=2^{n-1}$ 。目前实际主要使用单色屏或双色屏。

三、系统硬件

本系统硬件主要由上/下位机接口、显示屏、下

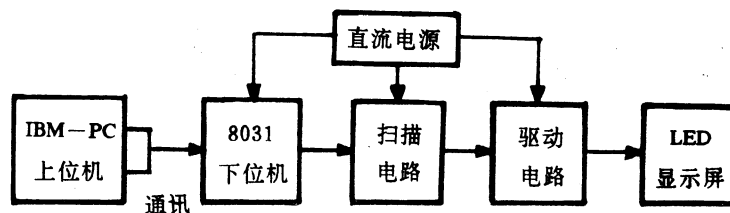


图 1 系统结构框图

位机控制等三部分组成。

1. 上、下位机接口部分

本大屏幕显示系统用一台 IBM-PC 作上位机,进行图文编辑和显示控制;用 8031 单片机系统做下位机,进行扫描和驱动。目前上、下位机的接口有串行、并行和视频信号通信三种。视频信号通信涉及编码与解码技术,必须配置一些专用硬件,在此不予讨论;串行通信大多采用 RS-232-C 标准,波特率最多为 9600bps (或 19200bps),难以实

现高速通信;因此,并行通信受到了重视。商品化的并行通信接口均采用 GPIB(即 IEE-488)标准,但为节约成本,兼顾实际需要,可不用专用芯片,而采用简化的通信协议,实现并行通信,具体接口见图 2。

图中的并行接口分为上位机并行接口卡和下位机数据接收器两部分。并行接口卡插在 IBM-PC 扩展槽内,在 IBM-PC 与下位机之间传送数据和控制信息。接口卡占用的口地址为 0x2f0~

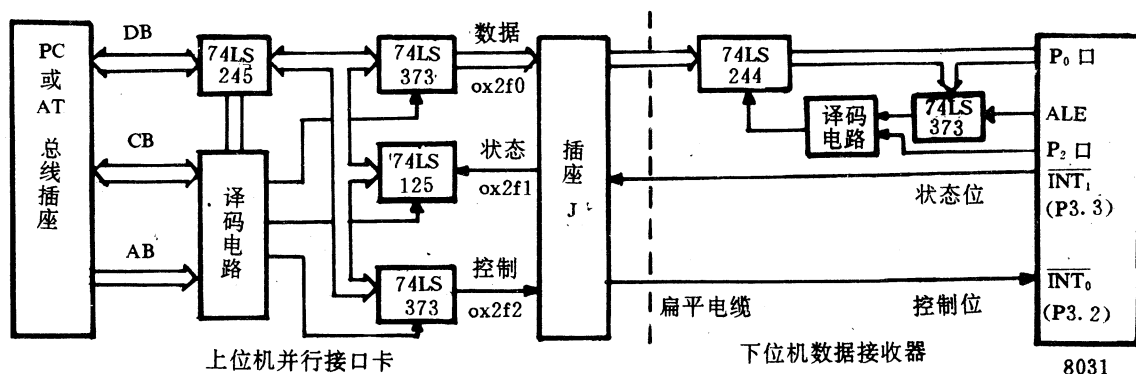


图 2 上、下位机并行通信接口

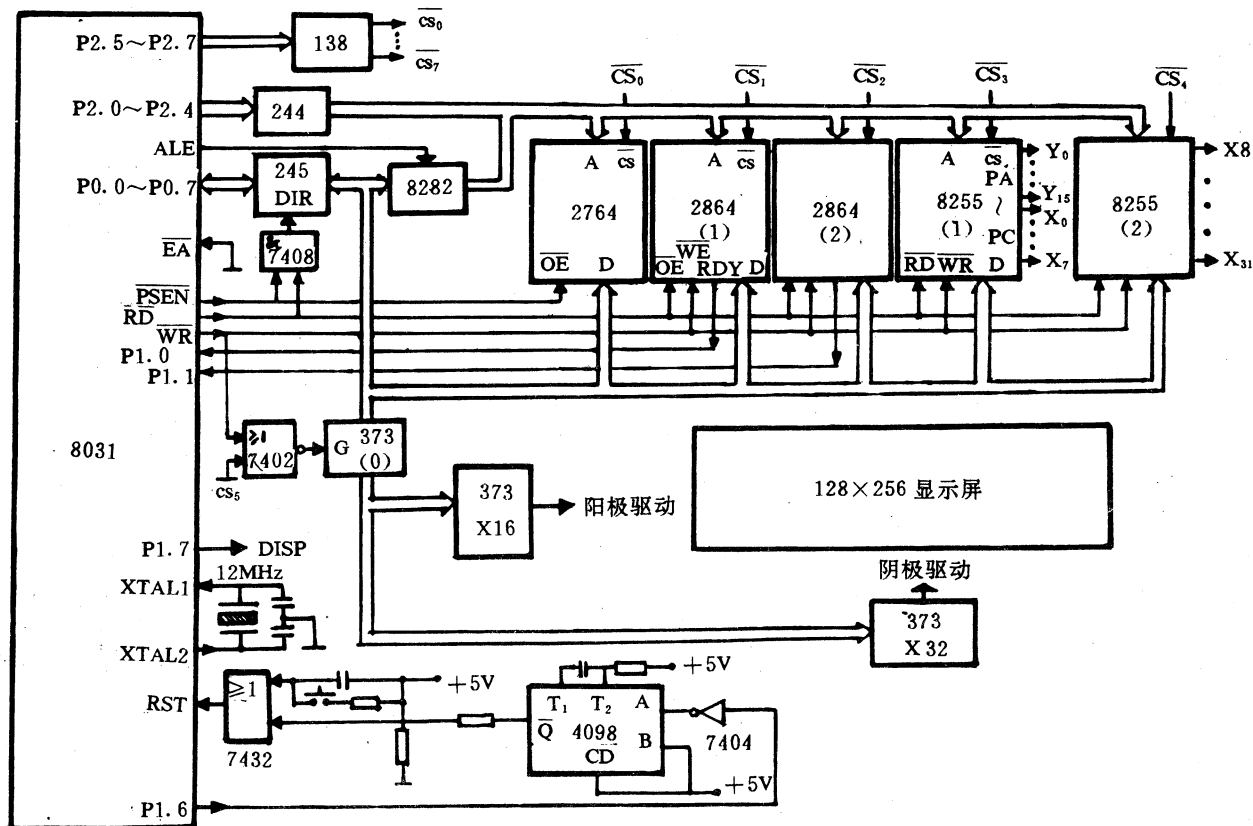


图 3 下位机显示控制电路框图

0x2f2, 其中 0x2f0 为写数据口 $D_0 \sim D_7$, 0x2f1 为读状态口, 0x2f2 为写控制口。数据接收器是下位机单片机系统的一部分, 它利用来自 IBM-PC 的控制信号作外中断源 $\overline{INT_0}$, 用一多功能口 $P_{3.3}$ 作状态位输出。它把来自 IBM-PC 的并行数据当作一个外部数据端口进行读操作, 下位机通过扁平电缆与并行接口卡上的插座 J 相连。并行通信要求扁平电缆线的长度应小于 15m。

2. 下位机显示控制部分

下位机显示控制电路框图见图 3。

8031 单片机是显示控制部分核心芯片。由于 P_0 口和 P_2 口低于 5 位外接器件较多,为了防止过载,它们分别采用双向驱动器 74LS245 和单向驱动器 74LS244 加以驱动。8282 锁存器用于把 P_0 口的地址低 8 位和数据分割开。 \overline{RD} 和 \overline{PSEN} 相与后接 74LS245 的方向控制端 DIR,以保证在读外部程序或数据存储器时,均能把数据经 P_0 口送入 8031,其它时间里 P_0 口输出驱动。

本系统扩展了两片 8KB 容量的 E²PROM

2864A, 它们用于存放需显示的汉字或图象数据信息。每片 2864 可存放两屏显示数据, 2864 可根据需要扩展到 4 片。2864 中内容由上位机 IBM-PC 经并行通信接口传送过来。2864 具有 RDY/ $\overline{\text{BUSY}}$ 功能, 即在数据写入 2864 期间, RDY/ $\overline{\text{BUSY}}$ 端输出处于低电平状态, 数据写入后, 输出电平变为高电平。8031 的 P 1.0、P 1.1 两引脚分别与两片 2864 的各个 RDY/ $\overline{\text{BUSY}}$ 端相连。8031 只要检测 P 1.0、P 1.1 电平情况, 就可判断数据写入 2864 的情况。2864 芯片本身具有掉电保护功能。

EPROM 2764 的作用是存放程序。74LS1383-8 译码器作用是根据 P_2 口高 3 位地址值,选通扩展的各有关芯片工作。8D 锁存器 74LS373(0)的作用是把 8031 输出的 8 位数据放大后送到 x、y 数据锁存器。两片 8255 的作用是选通 y 数据锁存器 74LS373(1)~74LS373(16)和 x 数据锁存器 74LS373(17)~74LS373(48)。当 8255 选通某个 74LS373 时,从 74LS373(0)输出的 8 位数据被送到此 74LS373 中。

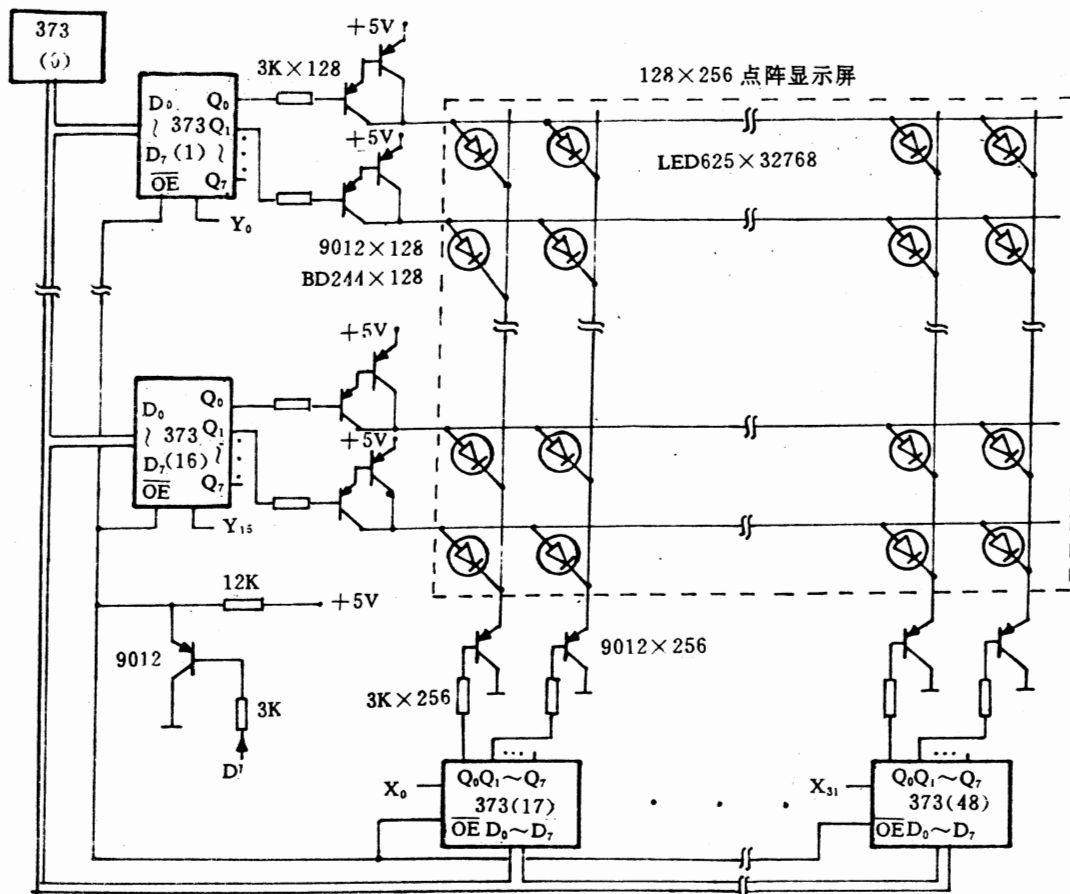


图 4 128×256 点阵显示屏及其行列数据锁存驱动电路

为防止尖峰等随机干扰引起系统“飞程序”，显示系统设计了看门狗电路。此电路由一单稳态电路CC4098实现，将它设计成脉冲漏失检出电路，电路按再触发方式连接。程序执行时，每隔 Δt 时间，单片机从P1.6脚发出一个负脉冲，适当调节阻容参数使单稳态输出脉宽 t_w 在 $\Delta t \sim 2\Delta t$ 之间。这样系统正常运行时，P1.6脉冲无丢失现象且 $t_w > \Delta t$ ，故 \overline{Q} 端输出为0。但当系统受干扰乱跳时，P1.6脉冲中有漏失，4098输出一正脉冲，强制系统复位。单片机上电/按键复位电路的输出与看门狗电路的输出经74LS32或门共同使单片机复位。

本系统晶振频率选为12MHz。因8031内部无ROM或EPROM，故EA脚应接地。P1.7脚用于控制显示屏显示或熄灭。

3. 显示屏及其锁存驱动部分

显示屏及其行列数据锁存驱动电路见图4。

目前LED发光二极管直径有 $\varnothing 3$ 、 $\varnothing 5$ 、 $\varnothing 8$ 等多种规格，它们主要区别在于控制电路中功率设计方面有差别。本显示屏为 128×256 点阵，每只LED采用 $\varnothing 5$ 高亮度发光二极管（如LED625等）。

由于静态显示每一个像素点需要一套驱动电路，而动态扫描显示则采用多路复用技术。每P个像素需一套驱动电路，P愈大驱动电路就愈少，成本也就愈低，引线也大大减少，更有利于高密度显示屏的制造，因此本显示系统采用动态扫描方式。动态扫描的关键是解决显示亮度和显示内容的稳定性。

由于人眼的视觉暂留现象的存在，当扫描期T小于约25ms时，人眼就感觉不到LED闪烁，看起来LED为连续、稳定发光。扫描周期T等于发光状态时间 τ_1 加上熄灭状态时间 τ_2 。 τ_1 增大，LED的亮度将增高。但本动态系统的扫描方式中， τ_1 的增大将受到扫描周期T、单片机执行指令的速度、扫描程序编制方法等因素的制约，且由于周期T时间内须扫描整个屏幕，按行扫描128行，而按列扫描则为256列，故 τ_1 必须小于 $T/128$ 或 $T/256$ 。此外，每个发光二极管的发光强度达到一定值后，会进入饱和状态，这时即使增加 τ_1 ，亮度也基本上不变。LED的饱和值由制造发光二极管的材料，工艺所决定。用于汉字显示器的LED最好选用高亮度型。本系统 τ_1 选为0.1ms。

本显示系统中，373(0)用于锁存总线上传输过来的显示内容或扫描信息，8255控制373(0)内容送扫描电路或显示内容锁存电路。

按行扫描时，373(0)~373(16)内锁存的是扫描信息，向其中任一芯片的输出端子送一个低电平

都可以点亮与该引脚端子相连的那一行LED发光二极管，依次从上到下传送一个低电平，就可顺序点亮整个屏幕上所有显示内容。373(17)~373(48)内锁存的是显示内容，即汉字或图像显示码。其每送一行点阵的显示代码，则373(1)~373(16)中某芯片与这一行LED点阵对应相连的输出端子输出显示有效低电平。接行下一行显示的内容送到373(17)~373(48)共256个输出端后，行扫描有效电平下移一行。这样每传送一次显示信息，就扫描与之对应的一行LED点阵，每次扫描一个循环即扫描完128行LED点阵，完成一次完整的屏幕内容显示。P1.7送出的DISP信号用于控制每行扫描有效电平时间为0.1ms。

按列扫描时，373(1)~373(16)与373(17)~373(48)两者中锁存的内容与按行扫描时情况相反，即前者内锁存显示内容，后者内锁存扫描信息。

LED发光二极管工作电压约1.0V，电流约15mA。按行扫描时，某一时刻同行的256个发光管同时发光其电流可达 $15\text{mA} \times 256 = 3.84\text{A}$ ；按列扫描时，某一时刻同列的128个发光管同时发光其电流为1.92A。本系列选用了驱动管9012和BD244两对管进行行驱动，9012单管进行列驱动，它们完全可达到系统驱动要求。

四、系统软件

本系统软件主要由上位机编辑处理程序、上位机发送程序、下位机中断接收程序、下位机主程序和下位机画面定时中断程序等组成。其中上位机编辑处理程序只需对IBM-PC配置的中文编辑软件稍作修改即可，具体不作介绍。

1. 上、下位机并行通信程序

由于8031晶振为12MHz，而80286/386/486晶振都高于20MHz，加上Cache等技术，所以IBM-PC工作速度远高8031。在此条件下，上位机只需检测下位机是否已准备好接收数据，若准备好则发送；否则等待。8031接收到数据后，将对应的状态位置或复位即可。

上位机发送数据之前，首先要通知8031准备接收。它通过写控制位来实现的。8031上电初始化及每次接收数据完毕后，都将状态位置为1，表示准备好状态。IBM-PC发出的控制字将引起8031的外中断，进入中断后8031将状态位置为0，表示忙状态。IBM-PC检测到此变化后即撤消控制位上的中断触发电平，并且转入数据发送程序，否则等待。数据通常以包的格式发送。包的格式应离线约定。为提高灵活性，可通过长度字、规定的结束符来定义变长包；为提高可靠性，包中定义分界符、检

验字、起止关键字等。数据发送期间,上位 IBM-PC 通过检测状态位是否已取反来判断是否可发送下一个数据。图 5 是上位机的发送软件框图。它可用 C 语言或汇编语言编写。

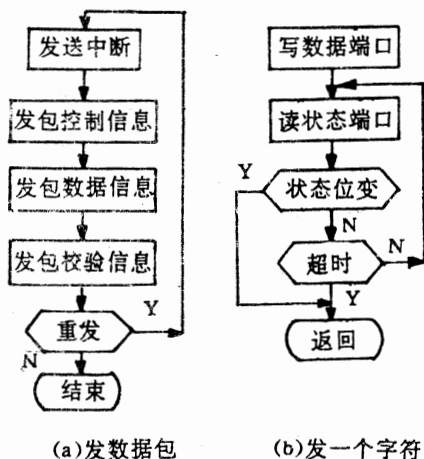


图 5 上位机发送程序框图

8031 上电复位后即初始化 $\overline{INT_0}$ 为外中断触发方式,并置状态位 $P_{3.3}$ 为 1。进入中断及接收每一个数据后,将状态位取反。如果数据未接收完,则延时片刻再读数据端口,接收完毕后,仍置状态位 $P_{3.3}$ 为 1,并返回主程序。图 6 给出了下位机 $\overline{INT_0}$ 中断接收程序框图。

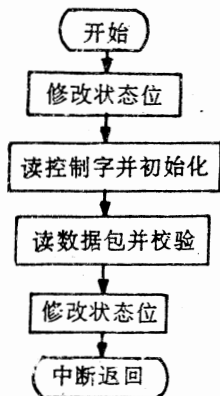


图 6 下位机 $\overline{INT_0}$ 中断接收程序框图

图 5 和图 6 稍作修改,就很容易实现一台 IBM-PC 与多个 8031 并行通信,实现多个显示屏同时显示。

2. 下位机显示控制程序

本系统扩展的 2 片 E^2PROM 可存放 4 幅画面(可再扩展 2 片 E^2PROM ,共 8 幅画面),选择显示画面子程序负责判断是显示第几个画面,从而决定从那个画面的存贮区取数据显示。定时器 T_1 是用来对整个画面在屏幕上停留时间进行定时。若时间到 10 秒则换下一个画面。由上位机传送来的每幅画面的显示方式存放在 8031 的内部 RAM 中,在每幅画面扫描显示之前,先取出其对应的显示方

式,根据显示方式不同,画面以不同样式显示。本系统设置了上移、下移、左平移、右平移、开幕式、合幕式和动画等 7 种显示方式。其中前 2 种采用按行扫描显示,只不过上移是从下向上扫描,下移是从上向下扫描而已。接着的 4 种都是按列扫

(下转 13 页)

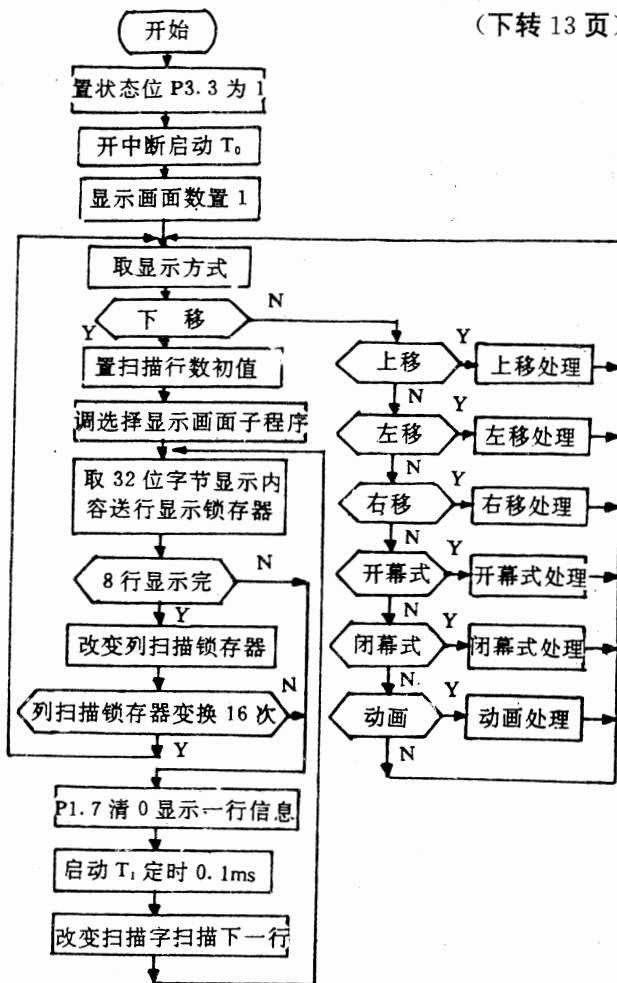


图 7 下位机显示主程序框图

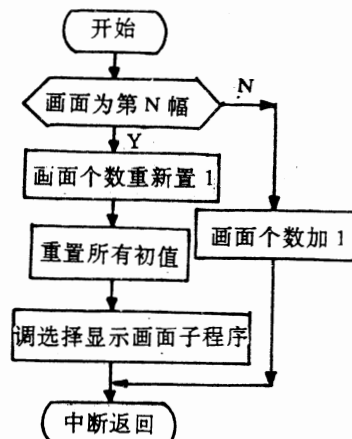


图 8 下位机 1 画面定时中断程序框图

灰色综合评估系统

国防科工委指挥技术学院[101407] 朱诗兵

摘要:本文以文献[1]所建立的灰色综合评判模型为依据,编写了解决多目标决策的灰色综合评估系统。

关键词:灰色综合评判模型,关联度,规格化

一、前言

在工程技术的设计或复杂系统的评价中,如何对备择方案进行优次排序或选择出最优对象,一直是系统工作者普遍感兴趣的课题,但“优”与“劣”的划分并不存在绝对清晰的界限,具有中介过渡性,这是客观存在的模糊性。近年来我国对此研究方兴未艾,有关的研究成果和论文很多。灰色综合评判模型是利用灰色关联度分析法、结合拓展的最小二乘准则来建立的,它能很好的处理贫信息系统。本系统是利用此模型为理论基础而编写的,它在解决多目标决策问题上起到了一定的作用。

二、模型的建立

1、属性矩阵

设系统有 n 个待评估对象,每个对象又有 m 个评估指标,评估对象在评估指标下的属性值构成如下属性矩阵:

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix} = (X_{ij})_{m \times n}$$

式中 X_{ij} 表示第 j 个评估对象在第 i 个评估指标下的属性值。

2、规格化矩阵

一般来说,对象的属性指标有如下三种,分别对其进行规格化处理:

(1)“效益型”(如产量、利润)

$$Y_{ij} = \frac{X_{ij} - \min_i X_{ij}}{\max_i X_{ij} - \min_i X_{ij}}$$

(2)“成本型”(如费用、成本)

$$Y_{ij} = \frac{\max_i X_{ij} - X_{ij}}{\max_i X_{ij} - \min_i X_{ij}}$$

(3)“适当型”即“越接近某一标准值 γ_i 越优”(如降雨量、土壤肥级等)

$$Y_{ij} = 1 - \frac{|X_{ij} - \gamma_i|}{\max_i |X_{ij} - \gamma_i|}$$

于是得规格化矩阵

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ \vdots & \vdots & & \vdots \\ Y_{m1} & Y_{m2} & \cdots & Y_{mn} \end{bmatrix} = (Y_{ij})_{m \times n}$$

显然, $0 \leq Y_{ij} \leq 1$, Y_{ij} 越大,表明第 j 个方案的第 i 个因素评价越优; Y_{ij} 越小,表明第 j 个方案的第 i 个因素评价越次。

我们分别定义系统的优向量 \vec{G} 和次向量 \vec{B} 如下:

$$\vec{G} = (g_1, g_2, \cdots, g_m) = (Y_{11} \vee Y_{12} \vee \cdots \vee Y_{1n}, Y_{21} \vee Y_{22} \vee \cdots \vee Y_{2n}, \cdots, Y_{m1} \vee Y_{m2} \vee \cdots \vee Y_{mn})$$

$$\vec{B} = (b_1, b_2, \cdots, b_m) = (Y_{11} \wedge Y_{12} \wedge \cdots \wedge Y_{1n}, Y_{21} \wedge Y_{22} \wedge \cdots \wedge Y_{2n}, \cdots, Y_{m1} \wedge Y_{m2} \wedge \cdots \wedge Y_{mn})$$

\vee, \wedge 分别为取大、取小运算符,又记:

$$\vec{Y}_j = (Y_{1j}, Y_{2j}, \cdots, Y_{mj}) \quad j = 1, 2, \cdots, n$$

3、关联度分析

灰色理论中的关联度分析,是一种新的衡量因素间关联程度大小的量化方法,是对系统统计数据列几何关系的一种比较。

向量 \vec{Y}_j 与优向量 \vec{G} 的关联系数为:

$$\zeta_i(\vec{Y}_j, \vec{G}) = \frac{\rho \max_i \max_j |Y_{ij} - g_i|}{|Y_{ij} - g_i| + \rho \max_i \max_j |Y_{ij} - g_i|}$$

向量 \vec{Y}_j 与次向量 \vec{B} 的关联系数为:

$$\zeta_i(\vec{Y}_j, \vec{B}) = \frac{\rho \max_i \max_j |Y_{ij} - b_i|}{|Y_{ij} - b_i| + \rho \max_i \max_j |Y_{ij} - b_i|}$$

上面的 ρ 为分辨系数, $\rho \in [0, 1]$, 一般取 $\rho = 0.5$ 。设评估指标的权重向量为 (至于建立,可采用层次分析法 AHP):

$$\vec{W} = (W_1, W_2, \dots, W_m), \sum_{i=1}^m W_i = 1$$

于是向量 \vec{Y}_j 与优向量 \vec{G} 的关联度为:

$$\gamma(\vec{Y}_j, \vec{G}) = \sum_{i=1}^m W_i \xi_i(\vec{Y}_j, \vec{G})$$

向量 \vec{Y}_j 与次向量 \vec{B} 的关联度为:

$$\gamma(\vec{Y}_j, \vec{B}) = \sum_{i=1}^m W_i \xi_i(\vec{Y}_j, \vec{B})$$

4、模型

假设第 j 个评估对象 \vec{Y}_j 以 μ_j 从属于优向量 \vec{G} , 那么 \vec{Y}_j 即以 $(1-\mu_j)$ 从属于次向量 \vec{B} . 为了建立系统的综合评估模型, 我们将经典最小二乘准则作合理拓展, 提出目标函数:

$$\min \{ F(\vec{\mu}) = \sum_{j=1}^n [(1-\mu_j)\gamma(\vec{Y}_j, \vec{G})]^2 + [\mu_j\gamma(\vec{Y}_j, \vec{B})]^2 \}$$

其中 $\vec{\mu}$ 为系统的最优解向量:

$$\vec{\mu} = (\mu_1, \mu_2, \dots, \mu_n)$$

由 $\frac{\partial F(\vec{\mu})}{\partial \mu_j} = 0$ 得:

$$\mu_j = \frac{1}{1 + \frac{\sum_{i=1}^m W_i \xi_i(\vec{Y}_j, \vec{B})^2}{\sum_{i=1}^m W_i \xi_i(\vec{Y}_j, \vec{G})^2}}$$

通过 $\mu_j (j=1, 2, \dots, n)$ 的大小比较, 我们即可对评估对象进行综合排序. 假设 $\mu_1 \geq \mu_2 \geq \dots \geq \mu_n$, 则评估对象的优次排序依次为: 对象 1 \geq 对象 2 $\geq \dots \geq$ 对象 n .

三、软件的研制

整个灰色综合评估通用软件包括五大部分, 其结构图如图 1.

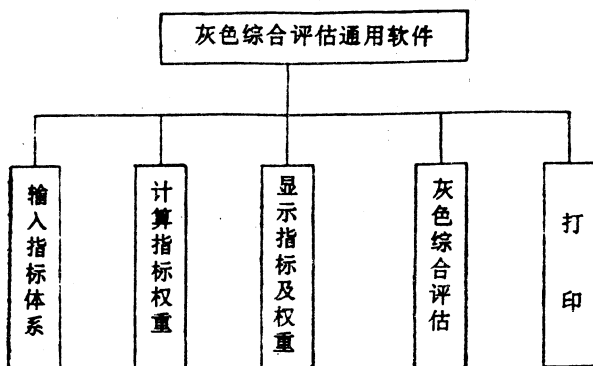


图 1

其中灰色综合评估模块的三个子功能模块如下(附部分程序): (input(int wight)函数是文本模式下的字符键盘接收函数)。

1、属性矩阵的规格化

key __x, key __y 在这里分别表示评估对象及评估指标的个数. Shuxing[][] 表示属性矩阵的各个属性值. Xmax[], Xmin[] 分别表示评估对象在评估指标下的最大指标属性值和最小指标属性值.

```

for(k=0; k<key __y; k++)
{
    printf("\n 请选择规格化公式: 1——效率型; 2——成本型; 3——适当型");
    do{
        sel=getch();
    }while((sel!='1')&&(sel!='2')&&(sel!='3'));
    if(sel=='1'){
        for(j=0; j<key __x; j++){
            Shuxing[k][j] = (double)(Shuxing[k][j] - Xmin[k]) / (Xmax[k] - Xmin[k]);
        }
    }
    if(sel=='2'){
        for(j=0; j<key __x; j++){
            Shuxing[k][j] = (double)(Xmax[k] - Shuxing[k][j]) / (Xmax[k] - Xmin[k]);
        }
    }
    if(sel=='3'){
        printf("\n 请输入标准值 r[%d]=", k);
        data=input(8);
        for(i=0; i<strlen(data); i++){
            ch[i]=data[i];
        }
        ch[i]='\0';
        r[k] = (double)atof(ch);
        for(j=0; j<key __x; j++){
            if(j==0) Xmax[k] = (double)fabs(Shuxing[k][j] - r[k]);
            if(Xmax[k] < fabs(Shuxing[k][j] - r[k]))
                Xmax[k] = (double)fabs(Shuxing[k][j] - r[k]);
        }
        for(j=0; j<key __x; j++){
            Shuxing[k][j] = (double)(Xmax[k] - fabs(Shuxing[k][j] - r[k])) / Xmax[k];
        }
    }
}
  
```

此时, Shuxing[k][j] 根据 k, j 的不同取值代表规格化矩阵中的各个属性值.

2、关联系数的计算

p 表示分辨系数, 一般取 $p=0.5$; w[] 表示评估指标的权重.

```

for(i=0; i<key __x; i++){
    for(j=0; j<key __y; j++){
        {
  
```



```

if (j == 0) r[i] = (double) fabs (Shuxing[j][i] - Xmax[j]);
if (r[i] < fabs (Shuxing[j][i] - Xmax[j]))
    r[i] = (double) fabs (Shuxing[j][i] - Xmax[j]);
if (j == 0) u[i] = (double) fabs (Shuxing[j][i] - Xmin[j]);
if (u[i] < fabs (Shuxing[j][i] - Xmin[j]))
    u[i] = (double) fabs (Shuxing[j][i] - Xmin[j]);
}
for (i = 0; i < key __ x; i++)
{
    if (r[0] < r[i]) r[0] = r[i];
    if (u[0] < u[i]) u[0] = u[i];
}
for (i = 0; i < key __ y; i++)
    for (j = 0; j < key __ x; j++)

```

```

{
    guanlian1[i][j] = (double) p * r[0] / (fabs (Shuxing[i][j] - Xmax[i]) + p * r[0]);
    guanlian2[i][j] = (double) p * u[0] / (fabs (Shuxing[i][j] - Xmin[i]) + p * u[0]);
}

```

此时, $guanlian1[][]$, $guanlian2[][]$ 分别表示评估对象与优向量的关联系数和评估对象与次向量的关联系数。

3、关联度及最优解向量的计算

```

for (i = 0; i < key __ x; i++)
{
    r[i] = 0;
    u[i] = 0;
    for (j = 0; j < key __ y; j++)
    {
        r[i] += (double) w[j] * guanlian1[j][i];
        u[i] += (double) w[j] * guanlian2[j][i];
    }
    uj[i] = (double) (1 / (1 + pow((double) u[i] / r[i], 2)));
}

```

此时, $r[]$, $u[]$ 分别表示评估对象与优向量的关联度和评估对象与次向量的关联度; $uj[]$ 表示最优解向量。

灰色综合评估模块的程序流程图如图2。

从上述程序流程图, 可以很容易了解灰色综合评估模型的建立过程。

四、结束语

文献[3]的实例, 依成绩—水平考试做“课程试卷质量的灰色综合评估”, 直接运行灰色综合评估系统, 在人机对话的情况下, 仅仅只须键入每个评价对象在相应各个评价因素下的属性值则可得课程试卷质量综合评估的排列名次。从其结果分析可看出灰色综合评估系统加大了数据的离散性, 提高了结果的可信性。

参考文献:

- [1] 罗小明等,《灰色综合评判模型》, 系统工程与电子技术, 1994年第9期
- [2] 邓聚龙,《灰色系统基本方法》, 华中理工大学出版社, 1987年
- [3] 买焕章等,《试卷质量的模糊综合评估》, 数理统计与管理, 1992年第2期
- [4] 王军政编,《Turbo C 2.0 实用高级编程技巧》, 北京科海培训中心, 1992年

(本文收稿日期: 1994. 12. 20)

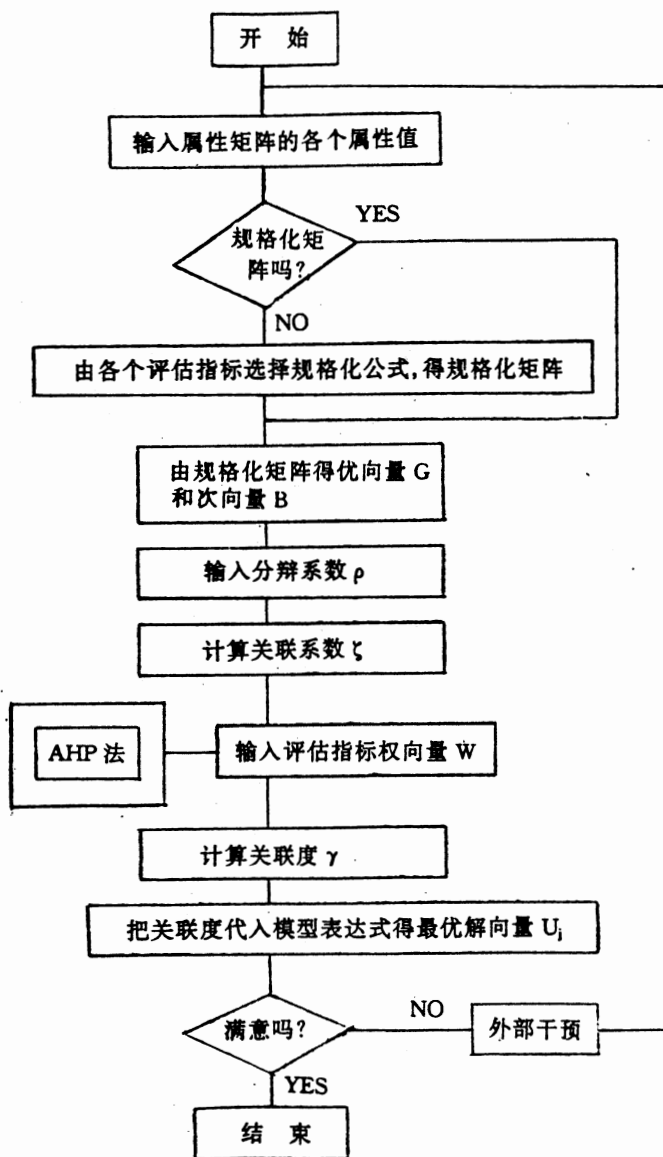


图2

计算机联网通信中调制解调器的研制

华南理工大学计算机系 肖南峰 林伟健

摘要 本文作者针对计算机与计算机,计算机与(控制)设备之间通过电话线路联网通信,设计了一种高性能的调制解调器(MODEM)。它可与串行通信接口(RS-232C)一起做成一块模板置于计算机内,用于企业管理自动化和生产自动化领域中的数据通信。

一. 引言

一些工厂、企业在实现计算机管理自动化或生产自动化时,往往要将地理位置分散的各种类计算机及(控制)设备联接起来,形成网络。但在有些场合,由于受各种环境及条件的制约,直接采用计算机(局部)网络难于达到目的,而借助于已有的电话网络将它们联接起来,则在目前是较有效的一种方法。

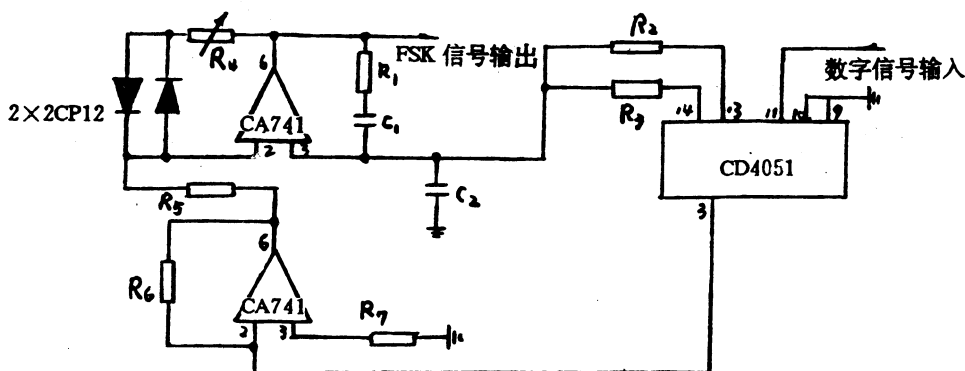
计算机及(控制)设备的数据信息要通过电话网络传输,必须进行调制解调,使数字信号变成音频信号传输,然后再恢复原数字信号。目前国内外厂家生产的用于话路数据通信的调制解调器普遍速率低、价格高、体积大。为此我们设计了一个用于计算机话路通信的 MODEM,它具有成本低、通信速率高、抗干扰能力强、性能稳定可靠等特点。

二. 电路设计

为使已有话路网络能照常工作,故电话线路应保持不变。本 MODEM 只设主信道,两线制,受 RS-232C 标准信号控制,频移键控(FSK),半双工,异步方式工作,整个 MODEM 主要由四个部分组成:调制器、解调器,与 RS-232C 标准的接口,强接电路。

(1) 调制器设计

它主要由振荡电路及频率转换开关组成。由于文氏(wien)电桥电路简单,振荡波形好,频率调节范围宽,且由于电路中电容具有惯性,当振荡器由一个频率转换成另一个频率时,相位基本上连续,可极大地方便解调,故选其作为调制器的振荡电路,另用两个 CA741 作运放,CD4051 模拟开关为频率转换开关,两个载频可按有关公式求得(参见参考文献 2)。见图一所示。

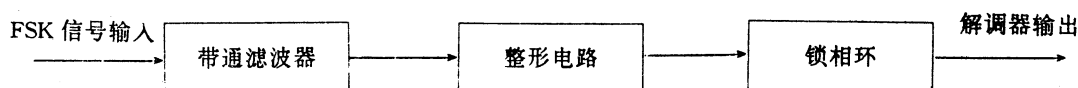


图一 调制器原理图

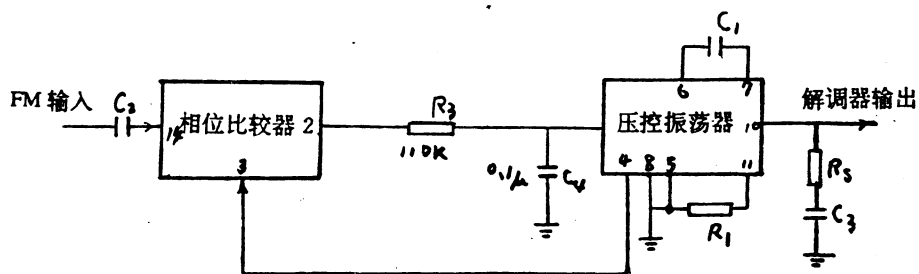
(2) 解调器设计

它主要包括带通滤波器,整形电路,锁相环等,见图二所示。带通滤波器是滤掉信道串入解调器的带外噪声,整形电路是将正弦波变成方波,同时切除与有用信号一起进入解调器的幅度干扰信号,其关键是设计锁相环。

我们选用适于制作音频段 FSK 解调器又便于与计算机接口联接的集成电路锁相环 CD4046。由于调频信号实际是一种频率突变信号,为使锁相环能跟上频率突变,必须采用二阶环,见图三所示,图中有关参数通过实验调整决定。



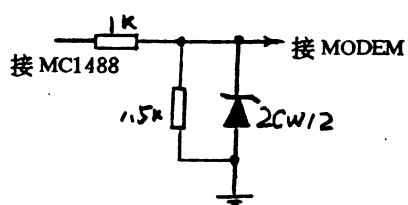
图二 解调器原理图



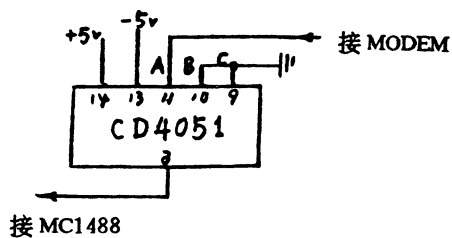
图三 锁相环原理图

(3) 与 RS-232C 标准的接口设计

按 RS-232C 标准,输出的串行数据信号是双极性信号,故本接口采用 MC1488,MC1489 分别作为串行口发送和接收的电平转换,由于 MC1488 输出电平为正负 10 伏电平,故采用如下转换电平电路以便控制频率转换开关 CD4051。另为防止串行口判断错误,产生误码,又采用另一转换电平电路,分别见图四(a)和(b)所示。



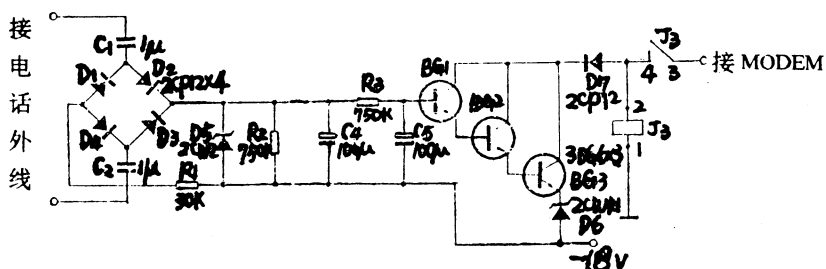
图四(a) 电平转换电路

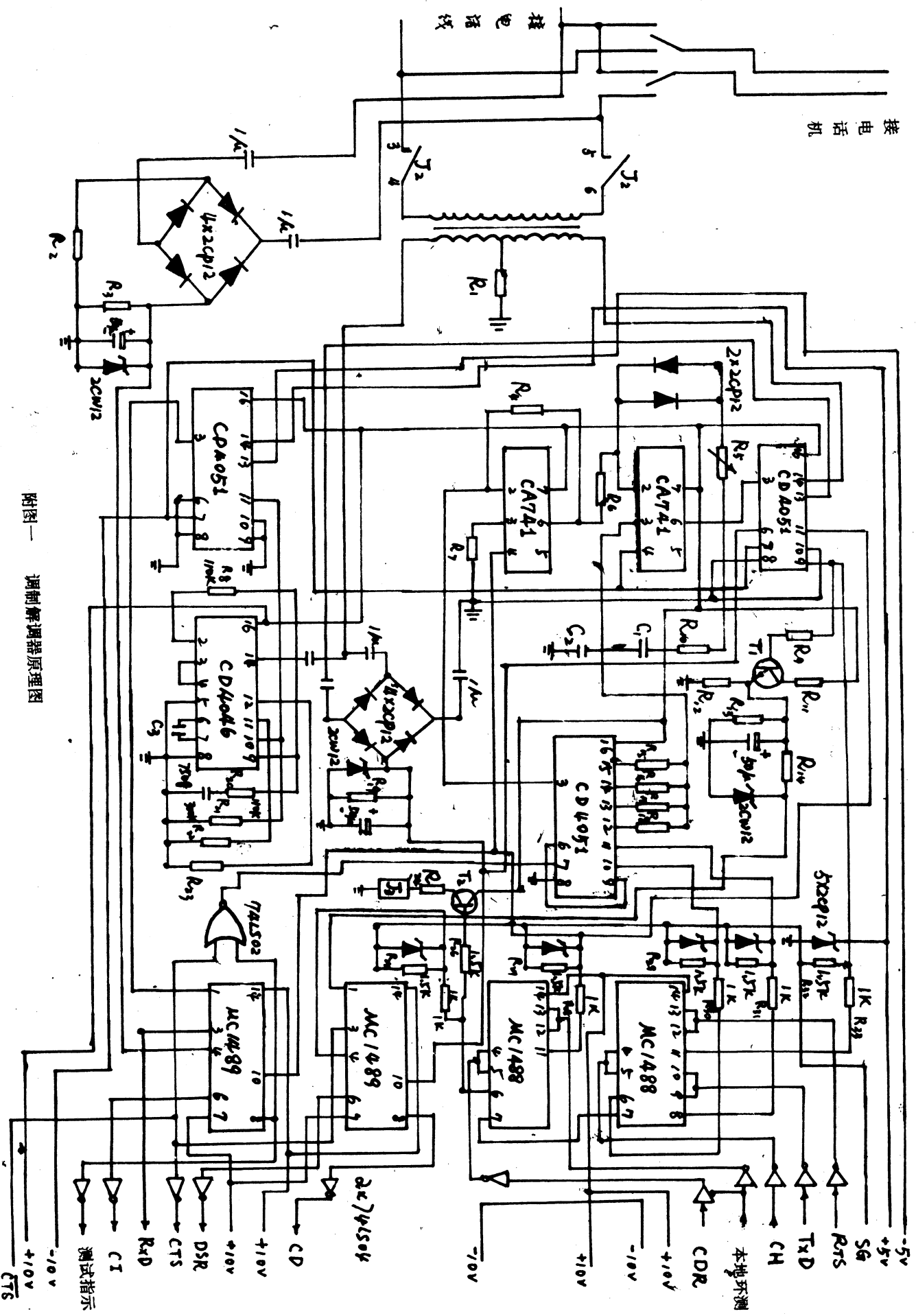


图四(b) 电平转换电路

(4) 强接电路设计

强接电路是接在电话外线的输入端。它的作用是,当数据接收方没有人时,可自动地将数据发、送双方的电路接通,发方可以发送数据,接收方可以自动地接收数据。其原理见图五所示。





附图一 调制解调器原理图

当发送方呼叫接收方时,由电话交换机送出3秒断续铃流。收方收到铃流经桥式全波整流($D_1 \sim D_4$),输出直流电压,向 C_4 电容器充电。当 C_4 电容器两端电压充到8~9.5伏时, D_5 稳压二极管击穿,使 C_4 两端电压稳定在8~9.5伏。 R_1 是限流电阻。 C_4 两端电压经 R_3 电阻向 C_5 充电, R_3C_5 组成迟动电路,其迟动时间为45~60秒钟, C_5 两端电压如果低于5.3伏时, BG_{1-3} 晶体管是处于截止状态, J_3 继电器不动,当 C_5 两端电压超过5.3~6.6伏时, G_{1-3} 开始工作,使 J_3 继电器动作。 BG_{1-3} 采用三管复合的目的,是为提高电流放大倍数,使其有足够大的集电极电流带动 J_3 继电器。

R_2C_4 组成迟复电路,其迟复时间为30~45秒。调整 R_2 电阻可以改变迟复时间。当 J_3 继电器动作后, J_3 接点接通交换机直流环路,停止铃流送出。这时 C_4 电容器两端电压经 R_2 电阻放电,30~45秒后其两端电压已低于5.3~6.6伏。这时 BG_{1-3} 截止, J_3 继电器复原,这个时间就是 J_3 继电器从吸动到释放的时间,即迟复时间,也就是接通

电路等候发送数据的时间,如果超过此时间未发送数据, J_3 继电器释放,电路自行切断,恢复原来状态。

三、结语

为使用方便,实际将该MODEM与串行接口合起来做成一块模板,置于IBM个人计算机插槽内,实验,证明其抗噪声性能好,通信速度快,维修方便,有推广价值,整个MODEM电路图见图五所示。

可以预见,计算机及(控制)设备通过电话网络进行联网数据通信,具有广阔前景。

参考文献

- [1]《中国集成电路大全 CMOS 集成电路》 国防工业出版社 1985年2月
- [2]王国定《CMOS 模拟集成电路的应用》 上海科学技术出版社 1985年11月

(本文收稿日期:1995.1.18)

大型LED汉字智能显示屏研制

(上接6页)

描。左平移是从右向左扫描,右平移从左向右扫描,开幕式从中间向两边扫描,合幕式从两边向中间扫描。最后1种动画比较复杂,它采用行列扫描混用实现。图7和图8为下位机程序框图。图中仅详细给出了下移扫描显示方式流程,其它方式略去。

五、结束语

大屏幕智能显示屏集电子、计算机、半导体等多项技术之大成,是高科技的结晶,它具有画面生动、操作简便、灵活可靠、显示内容易于更换等特点,目前本系统已投入批量生产,发展前景和应用范围广阔,社会经济效益明显。

参考文献

- [1]刘树东等,“PC与8031的并行通信”,《电子技术》,1994年5期
- [2]张艺,“LED智能显示屏的结构及驱动、显示电路原理”,《电子与自动化》,1994年1期
- [3]张有弘等,“基于8031串行口的LED电子广告牌”,《电子技术》,1993年11期
- [4]刘道贤等,“单片机控制的LED汉字显示器”,《电子技术》,1992年12期
- [5]李华等,《MCS-51系列单片机实用接口技术》,北京航

空航天大学出版社,1993

- [6]程启明,“主从集散型麦芽生产微机控制系统”,《工业仪表与自动化装置》,1993年3期
- [7]程启明,“硫化炉生产过程的主从微机监控系统”,《工业控制计算机》,1994年3期
- [8]程启明,“工厂生产管理智能显示屏研制”,《计算机应用》,1995年第3期

(本文收稿日期:1995.1.11)

本刊致订户、读者

本刊半年来收到大量的读者来信、来电,限于条件,未能一一答复,在此一并致谢。

现就发行、订购的有关问题补充说明如下:

1、凡在当地邮局订购的,若有意外收不到的或有缺损情况的,请及时与当地邮局交涉,直至向总发行:广东省报刊发行局反映。本刊社不直接向这些订户补偿。亦即来刊社打交道的,一律按零星邮购办理。

2、因错过订期,直接向本刊订购的,本刊服务部会认真负责办理。

3、限于条件,来函作特别咨询的读者请自负复函的邮资。

网络管理信息系统共享数据的冲突处理

广西大学 梁京章

摘 要:本文讨论了网络管理信息系统共享数据冲突处理。提出了解决冲突的基本原则和具体的解决方法。

关键词:MIS 网络 数据共享

在网络管理信息系统中,一个数据库不仅仅被某个子系统内的多个用户共享,而且往往被多个子系统所共享。在这种情况下,发生数据冲突是不可避免的。处理好共享数据的冲突问题,是开发网络管理信息系统能否成功的关键。因此,在开发网络管理信息系统时,除了要实现系统所要求的功能外,还要花相当大的精力在冲突处理上。本文以FOXBASE+为例,谈谈解决共享数据冲突的一些方法。

一、避免发生冲突

解决数据冲突最好的方法就是避免冲突。实际系统中大多数共享应用,只要在编程时遵循一定的规则,冲突都可以避免。这些规则主要有以下几点:

- 1、能不独占的尽量不要独占打开数据库。一个数据库被某个用户独占打开后,其他用户再企图打开(不管是独占方式还是共享方式),或者一个数据库被共享打开后,某个用户再企图独占打开,都将会发生冲突。除非确实需要,否则数据库都应共享打开。

- 2、必须以独占方式打开数据库时,占用时间应尽量短。如某次操作过程相当长,并不需要独占完成,只是到最后才需要使用一次独占操作命令(如PACK,ZAP等),在这种情况下,应先共享打开数据库,直到需执行独占操作命令时,再以独占方式重新打开数据库。

- 3、需要同时打开多个数据库并需进行互有关联的写操作时,应注意在确保全部数据库都已打开前,不要进行写操作,否则数据的完整性可能会受到破坏。

- 4、能用文件加锁解决的就不要独占。如INSERT BLANK,是要求独占数据库的,如果可能,应尽量用APPEND BLANK代替。

- 5、能记录加锁的就不要文件加锁。如

GET...READ,REPLACE命令。

- 6、尽量用编辑内存变量替代编辑字段变量。象GET...READ命令要求对记录加锁,如直接编辑字段,会长时间占据记录。采用先编辑内存变量,然后再锁住记录替换字段的值,再立即解锁的方法,就可使占据记录时间降为瞬时。

二、解决冲突的基本原则

- 1、独占操作优先于共享操作:象PACK,ZAP等要求独占数据库的操作,都可能使数据库的数据产生较大的变动。为了使数据能及时更新,这些操作应该比共享操作具有较高的优先权。当某一用户要求对数据库独占操作时,那些正共享使用该数据库的用户应尽快关闭数据库以便独占能进行。独占操作完成后,共享操作才能继续进行。

- 2、同类操作先操作者优先于后操作者:在独占操作与独占操作之间、共享操作与共享操作之间,后操作者如遇到冲突则只有等待,不能中断其他用户已经开始了的操作。

- 3、冲突发生需用户等待时,应在尽量不破坏原屏幕界面的前提下给出清晰的提示,并根据具体情况提供等待结束后自动恢复原有操作、在等待过程中中断当前操作返回菜单等功能给用户选择。

三、冲突的具体处理方法

- 1、出错处理程序:要处理冲突首先要捕捉到冲突。冲突实质上是一种出错,使用错误陷阱技术可很好地解决这一问题。发生冲突后,转入出错处理程序进行处理。不少文献和资料中都可找到出错处理程序的例子,这里不再多述。

2、独占操作中中断共享操作的方法

- ①设置一数据库独占标志数据库DZBZK(结构见后),在独占操作开始时,将相应数据库的独占请求标志置为".T.",该标志的作用一是通知正在共享使用该数据库的用户关闭数据库,以便保证独占操作的顺利进行,二是屏蔽掉其他用户对该数据

库的独占请求。

② 在共享操作程序的适当地方(通常是在等待键盘输入的地方)插入一段检测独占标志的程序,一旦检测到独占请求后马上转入独占处理程序(见 GXCL. PRG),该程序的主要功能是:显示提示信息、关闭数据库、检测独占操作是否已经完成、独占操作完成后重新打开数据库等。

③ 在独占操作程序中,每次独占打开数据库前,先进行如下处理(见 DZCL. PRG):检测数据库是否已被独占,如是则循环检测等待;置独占请求标志;重复打开数据库直至成功为止(因共享操作关库需一定时间,故独占打开不一定能一次成功)。

④ 独占操作完成后,应马上关闭数据库(如继续共享使用也应先关闭然后再共享打开),清除数据库的独占标志,以便其他用户继续共享使用。

Structure for database: C:\FOX\DZBZK.DBF

Number of data records: 14

Date of last update : 04/15/93

Field Field Name Type Width Dec

1 SJKM C 8

2 DZBZ L 1

** Total ** 10

* --- 数据库独占使用处理 DZCL. PRG --- *

PARAM KM_11

PRIV KM_11,S1,AA &&KM_11:需独占使用的数据库名

```

SELE 0
DO WHIL .T.
  @24,20 SAY SPAC(40)
  USE DZBZK
  LOCA FOR SJKM=KM_11
  IF DZBZ
    ?? CHR(7)
    @24,20 SAY '数据库已被独占,请等候或按Q键退出'
    A=INKEY(2)
    IF A=113.OR.A=81
      CLOS DATA
      RETU TO MAST
    ENDI
  ENDI
  IF RLOC()
    REPL DZBZ WITH .T.
  ENDI
  USE
  EXIT
ENDD

```

```

SELE &S1
USE &KM_11
* --- 加锁数据库 --- *
DO WHIL .T.
  IF FLOC()
    EXIT
  ELSE
    ?? CHR(7)
    @22,0 CLEA TO 24,79
    @23,16 SAY '====数据库正被其他用户使用===='
    @24,16 SAY '====请等候或按Q键退出===='
    A=INKEY(2)
    IF A=113.OR.A=81
      CLOS DATA
      SET COLO TO
      RETU TO MAST
    ENDI
  ENDI
  EXIT
ENDD
@22,0 CLEA TO 24,79
USE &KM_11 EXCL
RETU
* --- END OF DZCL. PRG --- *

```

* --- 共享处理程序 GXCL. PRG --- *

```

PRIV A,S1,DBF0,REC0,NDX0,I1,I,DBF1,NDX1
DIME DBF0(10),REC0(10),NDX0(10)
?? CHR(7)
?? CHR(7)
* --- 保存数据库状态 --- *
S1=LTRI(TRIM(STR(SELE(),2)))
I=1
DO WHIL I<=10
  I1=LTRI(TRIM(STR(I,2)))
  SELE &I1
  IF TRIM(DBF())<>'
    DBF0(I)=LTRI(TRIM(DBF()))
    NDX0(I)=LTRI(TRIM(NDX(1)))
  IF EOF()
    REC0(I)=0
  ELSE
    IF BOF()
      REC0(I)=-1
    ELSE
      REC0(I)=REC(N)
    ENDI
  ENDI
  SET INDE TO

```

```

GO BOTT
JLS0(I)=RECN()
ELSE
DBF0(I)=""
NDX0(I)=""
ENDI
I=I+1
ENDD
CLOS DATA && 关闭数据库以便其他用户独占
SET COLO TO
@22,0 CLEA TO 24,79
SET COLO TO GR+/B+
@23,25 SAY '数据库被独占使用,系统正常运行中断!!'
@24,25 SAY '=====请等候或按 Q 键退出
=====,'
* -----检测等待----- *
A=INKEY(5)
DO WHIL . T.
I=1
I2=0
DO WHIL I<=10
I1=LTRI(STR(I,2))
IF DBF0(I)<>""
DBF1=DBF0(I)
SELE &I1
ON ERROR I2=1
IF 'D: '$ DBF1
USE &DBF1 EXCL
ELSE
USE &DBF1
ENDI
ENDI
I=I+1
ENDD
IF I2=1
A=INKEY(1)
IF UPPE(CHR(A))='Q' && 返回菜单
CLOS DATA
RETU TO MAST
ENDI
ELSE
EXIT
ENDI
ENDD
* -----恢复打开数据库----- *
I=1
DO WHIL I<=10
I1=LTRI(STR(I,2))
SELE &I1
IF DBF0(I)<>""
DBF1=DBF0(I)

```

```

NDX1=NDX0(I)
IF NDX1<>""
* * * 如果是本地数据库则独占打开,下同 * * *
IF 'D: '$ DBF1
USE &DBF1 INDE &NDX1 EXCL
ELSE
USE &DBF1 INDE &NDX1
ENDI
ELSE
IF 'D: '$ DBF1
USE &DBF1 EXCL
ELSE
USE &DBF1
ENDI
ENDI
* -----如数据库记录数变化,退出当前模块返回
菜单----- *
IF RECC()<>JLS0(I)
?? CHR(7)
SET COLO TO
@22,0 CLEA TO 24,79
SET COLO TO GR+
@23,30 SAY '数据库变动较大,系统自动返回菜单!'
@24,30 SAY '=====如果需要,您可重新进入. =
=====,'
CLOS DATA
I1=INKEY(1)
SET COLO TO
CLEA
RETU TO MAST
ENDI
IF REC0(I)>0
GO REC0(I)
ELSE
IF REC0(I)=0
GO BOTT
SKIP
ELSE
GO TOP
SKIP -1
ENDI
ENDI
ELSE
USE
ENDI
I=I+1
ENDD
SELE &S1
RETU
* -----END OF GXCL. PRG----- *

```

(本文收稿日期:1994.1.3)

通用汉化菜单编制技术

浙江电视台 罗列异

摘 要: 本文结合监控系统菜单驱动模块的实例,详尽探讨了在 DOS 环境下开发通用多层下拉汉化菜单,论述菜单编制应注重通用性和扩充性。

关键词: 多层次下拉式,菜单,通用性,扩充性

一、概 述

几乎任何一个优秀软件都用到了菜单技术。目前在一些报刊资料中也经常介绍菜单编制技术,但多见的是单层下拉的,即使介绍多层下拉的,由于菜单驱动模块缺少相应的独立性,往往依赖于菜单项数、子菜单的层次等多方面因素,因此菜单通用性和扩充性较差,用户或软件开发人员在开发维护系统中不得不花费很大精力去修改菜单控制模块,以扩充系统功能或淘汰不必要的功能项。由此可见,编制一个通用性和扩充性强的菜单,可以起到事半功倍的效果。

二、菜单编制

本文介绍一种多层次下拉式菜单的编制。

菜单界面设计美观,下拉菜单项有阴影背景,多层菜单叠代,层次分明,立体感强;菜单驱动使用光标键(副键盘方向键亦可),带动光条,方便易行,并可从嵌套数层的子菜单直接跨越到另一组子菜单;汉字菜单(即使在西文 DOS 下),在屏幕上还有相应的菜单项的详尽描述,便于一般操作员使用,易学易会;在内存允许下,可以根据使用实际工作需要实现任意次下拉嵌套;菜单编制通用性强,易于修改及维护。

编制上述菜单,重要的是建立一个好的数据结构。我们定义

```
struct menu_str {
    int x0;      int y0;
    int count;   int len;
    char * head;
    char * content;
    char menu_str * * sub_menu;
    int ( * * funtion ) ();
    char * protect;
    char * * prompt;
}
```

数据结构 menu_str 各项具体含义是:

x0, y0 : 菜单左上角的座标 (X, Y)

count, len : 菜单宽度与项数

head : 主菜单的标题(子菜单该项为 NULL)
 content : 菜单项的内容,用二维指针表示
 sub_menu : 相应菜单项所对应的子菜单,若该项对应为函数则为空
 funtion : 相应菜单项所调用的函数,若该项对应为子菜单则为空;用二维函数指针表示
 protect : 相应菜单项的保护状态,值为 1 则处于保护状态
 prompt : 相应菜单项的详尽描述,构造成 LOTUS 风格的菜单项

基于上述数据结构 menu_str,我们设计一种递归算法来实现菜单多层次下拉式控制。程序运行中,当选择第一层子菜单,也就进入到递归算法中。如果用户按回车键,便可进入相应的子菜单中,即递归一次,最终到相应的选项对应为函数体为止;如果按 ESC 键,则退出一层递归,一直可以退至递归起点,即主菜单 Bar 时止;如果使用 ←→ 光标键,则控制模块会自动逐层退出递归体,然后进入到另一组子菜单递归算法中。由于整个菜单控制模块是相对独立的,我们无论在开发新的软件或是维护系统时,仅需对菜单项的内容作一些修改,然后添加所需新的功能模块,删除旧的模块,而不需要对菜单控制模块本身进行修改,因而通用性和扩充性很强。

本菜单控制模块系我们开发的监控系统重要组成部分,原程序已在 Borland C++ 3.1 编程通过,并在 NOVELL 网, DOS 5.0, 汉字 SPDOS 5.2 下运行正常。

程序所用的字库系 SPDOS 5.X 下 cclibj.dot 软字库。由于各种汉字位图字模在字库存取规律不尽相同,因此各种字库有时会不兼容。有些报刊介绍的 WPS 2.X 版系统位图字库读取方法是不全面的。经过实践和分析,我们得出几种常见汉字库读取方法:

(一) 16 * 16 位图简体

1~9 区汉字字模数据起始地址

$$= [(高字节 - 0XA1) * 94 + (低字节 - 0XA1)] * 32L \quad (式 2.1)$$

16~87 区汉字字模数据起始地址

$$= [(高字节 - 0XA7) * 94 + (低字节 - 0XA1)] * 32L \quad (式 2.2)$$

以上适用于软字库 cclibj.dot, xsdos.lph 和 cclib。

UCDOS 3.0 版 Hzk16 软字库 1~87 区用统一公式 (式 2.1)。

(二) 24 * 24 位图

16~87 区一、二级汉字字模数据起始地址

$$= [(高字节 - 0XA1) * 94 + (低字节 - 0XA1)] * 72L \quad (式 2.3)$$

以上适用于 UCDOS 软字库 Hzk24 点阵。

软字库 xsdos.lph 为 (式 2.3) + 304,560

cclib24 为 (式 2.3) + 18,432

值得注意的是 16 * 16 位图字模是按显示格式排列的, 而 24 * 24 位图字模是按打印格式排列的, 在编程时应有所不同。

三、菜单驱动模块程序清单

```
# include "stdio.h"
# include "conio.h"
# include "graphics.h"
# include "fcntl.h"
# include "process.h"
# include "dos.h"
# include "alloc.h"
# include "string.h"
# include "stdlib.h"
# include "bios.h"
# include "mkey.h"
// 宏定义菜单的位置, 以便于移动
# define HEAD_Y 10
# define MAIN_Y 70
# define SM_Y 190
# define SM_X 48
# define MAIN_C WHITE
# define HEAD_C WHITE
# define MENU_C WHITE
# define BK_C 11
// 菜单数据结构定义
struct menu_str {
    int x0; int y0;
    int count; int len;
    char * head;
    char * * content;
    struct menu_str * * sub_menu;
    int (* * function)();
    char * protect;
    char * * prompt;
};
```

// 输入键的联合

```
union KEYBOARD { char ch[2];
    int i;
};
```

// 函数说明

```
static int (* c1[8])() = {mes, no, help, esp_send, exit_s,
    no, prn, graph_mes},
    (* c2[4])() = {...},
```

// 菜单项的内容指针数组

```
static char * a1[8] = {"版权信息", "站况编辑", "帮助信息",
    "专访命令", "退出系统", "系统配置", "打印服务", "自动控制"};
```

static char * a2[4] = {...};

// 菜单项所调用的函数说明

```
static struct menu_str clock = {60, 210, 8, 8, "", a1, b11,
    c11},
```

```
* b1[8] = {NULL, &station, NULL, NULL, NULL,
    &clock, NULL, NULL},
```

* b2[4] = {...};

// 主菜单说明

```
static struct menu_str frame[6] = {
    {25, 140, 8, 7, "系统操作", a1, b1, c1},
    {...},
```

int handle_hzk;

// 以下为书写汉字的子函数, 包括书写 ASCII (函数能自动变实)

char far * cp = (char far *) 0xf000fa6e;

int getbit(unsigned char c, int n)

```
{ return ((c >> n) & 1);
}
```

int puthz (int x, int y, int z, int color, char * p, int number)

```
{ unsigned char i, c1, c2, f = 0, l1 = 1;
```

int i1, i2, i3, rec;

long l;

char by[32], ch;

while ((i = * p++) != 0)

```
{ if (i < 0x80) { l = i * 8;
```

for (i1 = 0; i1 < 8; i1++)

for (i2 = 0; i2 < 8; i2++)

if (getbit (* (cp + l + i1), 7 - i2))

putpixel(x + i2, y + 4 + i1, color);

x = x + z + 8;

if (++l1 > number) return (x);

}

if ((f == 0) && (i > 0x80))

```
{ if (i > 0xaa)
```

```
{ c1 = (i - 0xa7) & 0x07f;
```

f = 1;

}

else if ((i < 0xab) && (i > 0x80))

```
{ c1 = (i - 0xa1) & 0x07f;
```

f = 1;

}

else if (f == 1) {

c2 = (i - 0xa1) & 0x7f;

f = 0;

rec = c1 * 94 + c2;

```

l=rec * 32L;
lseek(handle_hzk,l,SEEK_SET);
read(handle_hzk,by,32);
for(i1=0;i1<16;i1++)
    for(i2=0;i2<2;i2++)
        for(i3=0;i3<8;i3++)
            if(getbit(by[i1 * 2+i2],7-i3))
                putpixel(x+i2 * 8+i3,y+i1,color);
x=x+z+16;
if(++l1>number) return(x);
}
return(x);
}
// 绘制菜单的阴影
void draw_shaw(int x0,int y0,int x1,int y1,int c0)
{
    setfillstyle(SOLID_FILL,c0);
    bar(x0,y0,x1,y1);
    setfillstyle(SOLID_FILL,8);
    bar(x1+1,y0+12,x1+12,y1+12);
    bar(x0+18,y1+1,x1+12,y1+12);
}
// 绘制菜单的框架
void fra(int x0,int y0,int x1,int y1)
{
    setcolor(8);
    rectangle(x0,y0,x1,y1);
    rectangle(x0+3,y0+3,x1-3,y1-3);
    setcolor(7);
    rectangle(x0+1,y0+1,x1-1,y1-1);
    rectangle(x0+2,y0+2,x1-2,y1-2);
}
// 绘制带阴影的菜单
void make_shaw(int x0,int y0,int x1,int y1,int c0)
{
    draw_shaw(x0,y0,x1,y1,c0);
    fra(x0,y0,x1,y1);
}
// 绘制窗口的标题
void barmes(int x0,int y0,int x1,int z,char * p,int color)
{
    setfillstyle(SOLID_FILL,color);
    bar(x0,y0,x1,y0+23);
    setfillstyle(SOLID_FILL,7);
    bar(x0+1,y0+1,x0+22,y0+22);
    setfillstyle(SOLID_FILL,8);
    bar(x0+4,y0+11,x0+20,y0+13);
    puthz(x0+32,y0+3,z,8,p,50);
}
// 绘制阴影的窗口
void make_window(int x0,int y0,int x1,int y1,int c,char
    * p,int z)
{
    make_shaw(x0,y0,x1,y1,c);
    barmes(x0+5,y0+6,x1-5,z,p,YELLOW);
}
// 交互式信息提示处理
int prompt(char * p,int f,int c)
{
    long i;
    char ok;

```

```

    char color,* p1,color1=LIGHTMAGENTA,color2=
    WHITE;
    union KEYBOARD key_get;
    i=imagesize(120,180,520,290);
    p1=(char *) malloc(i);
    if(p1==NULL) return 3;
    getimage(120,180,520,290,p1);
    make_shaw(120,180,500,270,c); /* c is the col-
    or of the window */
    puthz(150,195,2,8,p,50);
    ok=0;
    if(f==1) /* if exit then ok or cancel */
    {
        do { draw_shaw(180,228,280,250,color1);
            draw_shaw(340,228,440,250,color2);
            puthz(200,232,2,LIGHTBLUE," 确认 ",20);
            puthz(360,232,2,LIGHTBLUE," 取消 ",20);
            while(bioskey(1)==0);
            key_get.i=bioskey(0);
            if((key_get.ch[1]==75)|| (key_get.ch[1]==
            77))
            {
                if(ok) ok=0;
                else ok=1;
                color=color1;
                color1=color2;
                color2=color;
            }
        } while(key_get.ch[0]!=13);
        putimage(120,180,p1,COPY_PUT);
        free(p1);
        if(ok==0) return 0; /* ok return 0; */
        else return -1;
    }
    draw_shaw(260,228,360,250,color2);
    puthz(280,232,2,LIGHTBLUE," 确认 ",20);
    getch();
    putimage(120,180,p1,COPY_PUT);
    free(p1);
    return 0;
}
// 主程序开始
main ()
{
    int gdriver=DETECT,gmode;
    handle_hzk=open("mw.dot",O_RDONLY|O_BINA-
    RY);
    if( handle_hzk == -1) { printf(" ER1: Error on
    open CCLIB.dot !");
        exit(1);
    }
    registerbgidriver(EGAVGA_driver);
    initgraph(&gdriver,&gmode,"");
    if ( gdriver!= HERCMONO&&gdriver!=
    VGA&&gdriver!=EGA)
    {
        closegraph();
        printf(" ER2 : Error on the HERC or VGA EGA
        adapt");
        getch(); exit(2);
    }
}

```

```

directvideo=1;
mode=getgraphmode();
maxy=getmaxy();
box();
make_main();
closegraph();
close(handle_hzk);
return 0;
}
// 绘制主菜单的标题
void make_head()
{
    make_shaw(10, HEAD_Y, 620, HEAD_Y + 32,
    HEAD_C);
    barmes(14, HEAD_Y + 4, 616, 2, " 数字微波监控系统", HEAD_C);
    setcolor(8);
    settxtstyle(TRIPLEX_FONT, HORIZ_DIR, 2);
    outtextxy(310, HEAD_Y + 4, " DIGITAL MICROWAVE MONITOR ");
}
// 绘制主菜单的框架
void box()
{
    int i;
    make_head();
    make_window(10, MAIN_Y, 620, maxy - 56, MAIN_C, "主菜单窗口", 10);
    make_window(MES_X, MES_Y, 400, maxy - 110, MES_C, "信息窗", 20);
    setcolor(8);
    rectangle(20, MAIN_Y + 35, 610, MAIN_Y + 55);
    for(i=0; i<6; i++)
        puthz(30 + 590/6 * i, MAIN_Y + 37, 0, 8, frame[i].
head, 10);
}
// 主菜单控制
void make_main()
{
    int l;
    union KEYBOARD key_get;
    num=0;
    l=30+590/6;
    size=imagesize(l-8, MAIN_Y+37, l+72, MAIN_Y+55);
    buffer=(char *)malloc(size);
    barmes(15, MAIN_Y+5, 615, 10, "主菜单窗口", LIGHTMAGENTA);
    do { while (bioskey(2)==0x08);
        do { size=30+590/6 * num;
            getimage(size-8, MAIN_Y+37, size+72, MAIN_Y+55, buffer);
            setfillstyle(SOLID_FILL, LIGHTBLUE);
            bar(size-8, MAIN_Y+37, size+72, MAIN_Y+55);
            puthz(size, MAIN_Y+37, 0, YELLOW, frame[num].head, 10);
            while(bioskey(1)==0);
            key_get.i=bioskey(0);
            /* receive the message */

```

```

            putimage(size-8, MAIN_Y+37, buffer, COPY_PUT);
            if(key_get.ch[1]==LEFT) num=num==0?
5:--num;
            if(key_get.ch[1]==RIGHT) num=num==
5? 0:++num;
            } while(key_get.ch[1]!=DOWN&&key_get.ch
[0]!=TAB);
            // TAB 键用于切换到其它控制窗口
            else { do
                { ceng1=1;
                    flag=0;
                    size=30+590/6 * num;
                    getimage(size-8, MAIN_Y+37, size+72, MAIN_Y+55, buffer);
                    setfillstyle(SOLID_FILL, LIGHTBLUE);
                    bar(size-8, MAIN_Y+37, size+72, MAIN_Y+55);
                    puthz(size, MAIN_Y+37, 0, YELLOW, frame[num].head, 10);
                    make_menu(frame[num]);
                    } while ((flag!=2)&&(flag!=10)&&(ceng1>0));
                }
                fflush(stdin);
            } while( flag!=2);
            free(buffer);
        }
    // 以下子函数构造递归的算法
    int make_menu(struct menu_str mo)
    {
        int x0=mo.x0, len=mo.len, count=mo.count;
        int y0=mo.y0, size1=i, key1, key2, k;
        char *buf0, *buf1;
        int endx=x0+16+len*16;
        int endy=y0+16+count*20;
        size1=imagesize(x0, y0, endx+12, endy+12);
        buf0=(char *)malloc(size1);
        if(buf0==NULL) { prompt("内存不够.. 按任意键继续!", 0, LIGHTRED);
            return 1;
        }
        getimage(x0, y0, endx+12, endy+12, buf0);
        make_shaw(x0, y0, endx, endy, WHITE);
        for(i=0; i<count; i++)
            if(mo.protect[i])
                puthz(x0+12, y0+8+20*i, 0, 1, mo.content[i], 7);
            else
                puthz(x0+12, y0+8+20*i, 0, 1, mo.content[i], 10);
            i=0;
            size1=imagesize(0, 0, 16*len, 16);
            buf1=(char *)malloc(size1);
            getimage(x0+10, y0+8+20*i, endx-8, y0+24+20*i, buf1);
            setfillstyle(SOLID_FILL, LIGHTBLUE);
            bar(x0+10, y0+8+20*i, endx-8, y0+24+20*i);
            puthz(x0+12, y0+8+20*i, 0, YELLOW, mo.con-

```



```

tent[i],10);
do { while(bioskey(1)==0);
    key1=key2=bioskey(0);
    key1=key1&0xff;
    key2=key2&0xff? 0:key2>>8;
    if(key2==UP|key2==DOWN)
    { putimage(x0+10,y0+8+20*i,buf1,COPY_
PUT);
    if(key2==DOWN) i=i==count-1? 0:++i;
    if(key2==UP) i=i==0? count-1:--i;
    getimage(x0+10,y0+8+20*i,endx-8,y0+
24+20*i,buf1);
    setfillstyle(SOLID_FILL,LIGHTBLUE);
    bar(x0+10,y0+8+20*i,endx-8,y0+24+20
*i);
    if(mo.protect[i])
    puthz(x0+12,y0+8+20*i,0,1,mo.content[i],
7);
    else
    puthz(x0+12,y0+8+20*i,0,1,mo.content[i],
10);
    }
    if(key1==ESC)
    { switch (cengl) {
    case 0 : exit_s();
        break;
    case 1 : size=30+590/6*num;
        putimage(size-8,MAIN_Y+37,buffer,COPY_
PUT);
    default : cengl--;
        putimage(x0,y0,buf0,COPY_PUT);
        free(buf0);
        free(buf1);
        return(3);
    }
    }
    if(key2==LEFT||key2==RIGHT)
    { putimage(x0,y0,buf0,COPY_PUT);
    free(buf0);
    free(buf1);
    flag=1;
    size=30+590/6*num;
    putimage(size-8,MAIN_Y+37,buffer,COPY_
PUT);
    if(key2==LEFT) num=num==0? 5:--num;
    /* if left */
    if(key2==RIGHT) num=num==5? 0:++
num; /* if right */
    return(1);
    }
    if(key1==ENTER&&mo.protect[i])
    { if(mo.sub_menu[i]!=NULL)
        { cengl++;
        k=make_menu(*mo.sub_menu[i]);
        }
    else {
    position=i;

```

```

    k=( * mo.function[i])();
    switch(k) {
    case -1: prompt("系统错误! 按任意键继续..",0,
LIGHTRED;
    case 2 : flag=1;
        break; /* k=-1 is the error code =-1
;k=2 is a flag to redraw the box */
    }
    kb_act=0;
    }
    }
    if(key1==TAB)
    { flag=10;
    size=30+590/6*num; }
    // TAB 键用于切换到其它控制窗口
    } while(flag==0);
    putimage(x0,y0,buf0,COPY_PUT);
    free(buf0);
    free(buf1);
    return(0);
    }
    // 退出系统, 设置宏标志
    int exit_s()
    { int i;
    i=prompt("您是否真想退出系统?",1,BK_C);
    if(i==0) flag=2;
    return 0;
    }
    // 版本信息窗
    int mes()
    { char *buffer,i;
    long size;
    size=imagesize(140,150,515,335);
    buffer=(char *) malloc(size);
    if(buffer==NULL) return -1;
    getimage(140,150,515,335,buffer);
    make_window(140,150,500,320,LIGHTGREEN,"版
权窗口",10);
    puthz(220,190,6,BLUE,"微波通信服务系统 V1.0",
15);
    puthz(260,248,5,LIGHTBLUE,"1994年6月",15);
    draw_shaw(280,275,380,295,7);
    puthz(300,277,20,RED,"确认",3);
    getch();
    putimage(140,150,buffer,COPY_PUT);
    free(buffer);
    return 0;
    }

```

参考文献:

- [1] 孙晓敏,《通用菜单制作程序》,《电脑与微电子技术》,1993年第2期,20页
- [2] 尹彦芝,《C语言高级实用教程》,清华大学出版社,1993,5

(本文收稿日期:1994.12.20)

用图形成像技术生成各种新颖立体菜单

测井研究所 晁永胜

摘 要:本文讲述用图形成像技术生成新型立体菜单的方法

关键词:RGB 图形成像 立体菜单

一个软件的生命力,很大程度上借助于它的外观设计,用户界面的优劣直接影响着软件的成功与否,在众多的用户界面中,菜单驱动以其直观、简单而著称,在市场上占据着主要地位。随着应用的发展,要求菜单制作技术不断推陈出新满足用户需求。下面介绍一下利用图形成像技术生成的几种立体菜单,这几种菜单制作简单、使用方便、立体感强。

制作这几种菜单的一个关键技术源于图形成像,它通过把红(RED)、绿(GREEN)、蓝(BLUE)三种基本色按比例调配成深浅明暗不同的混合色,然后通过一定的算法配合在一起,便形成了精美的立体菜单。如白色(WHITE)就是由红、绿、蓝按1:1:1比例配制而成的,其中三种基本色的数值越大,颜色的亮度越高。本例中用RGB即红绿蓝共调配了9种颜色,其中颜色3、4、5、6配方分别为:05:2a:3f,05:0a:20,05:15:2a,05:1a:2f,按亮度顺序(升序)排列为:4、5、6、3,其配制方法使用C语言中的remappalette语句,颜色1、2、7、8、15配方为:2a:00:00,00:2a:00,2a:2a:2a,15:15:15,3f:3f:3f。1即蓝色,2即绿色,7为白色,8为灰色,15为亮白色,亮度顺序 $8 < 7 < 15$ 。

一、CubMenu0()凹式立体菜单

它是把用RGB调好的颜色,按从亮到暗的次序从外到里依次用画框、画线功能实现,这样生成的菜单从外到里,亮度逐渐减少,就象一个凹陷的物体,立体感十分强烈。具体程序见CubMenu0()。

二、CubMenu1()凸式立体菜单

它的制作基本上和凹式菜单相反。它是把RGB调好的颜色按从暗到亮的次序依次从外向里分别用画框、画线功能实现,这样生成的菜单亮度逐渐增加,就象一个凸面体。具体程序见CubMenu1()。

三、Bar3d()实物立体菜单

它是由一个长方体构成,正、侧面分别为明暗不同的颜色所填充,看起来很象一个棱角突出的实

心盒子。具体程序见Bar3d()。

四、Shadow()立体投影式菜单

它是由两个填充后部分重叠的矩形构成,第一个矩形的颜色较亮,第二个矩形与第一个矩形部分重叠,颜色较暗,看起来象第一个矩形的投影。具体程序见Shadow()。

上述程序由MSC 6.0编制,并在Compaq 386上调式通过,而且运行效果很好,可以直接为用户使用。

```
#include <stdio.h>
#include <graph.h>
#include <conio.h>
main()
{
    setvideomode( VRES16COLOR );
    remappalette(3,0x052a3fL);
    remappalette(4,0x050a20L);
    remappalette(5,0x05152aL);
    remappalette(6,0x051a2fL);
    remappalette(1,0x2a0000L);
    remappalette(2,0x002a00L);
    remappalette(7,0x2a2a2aL);
    remappalette(8,0x151515L);
    remappalette(15,0x3f3f3fL);
    CubMenu0(5,5,200,200);
    CubMenu1(250,95,500,150);
    setcolor(1);
    Bar3d(5,270,250,470,10);
    Shadow(300,250,500,470);
    getch();
    setvideomode( DEFAULTMODE );
}

int CubMenu0( int x,int y,int xx,int yy)
{
    setcolor(3);
    rectangle(_GBORDER,x,y,xx,yy);
    rectangle(_GBORDER,x,y+1,xx-1,yy);
    setcolor(4);
    moveto(x,y);
    lineto(x,yy);
    lineto(xx,yy);
```

```

moveto(x+1,y);
lineto(x+1,yy-1);
lineto(xx,yy-1);
setcolor(5);
rectangle(_GBORDER,x+2,y+2,xx-2,yy-2);
rectangle(_GBORDER,x+3,y+3,xx-3,yy-3);
rectangle(_GBORDER,x+4,y+4,xx-4,yy-4);
rectangle(_GBORDER,x+5,y+5,xx-5,yy-5);
setcolor(4);
rectangle(_GBORDER,x+6,y+6,xx-6,yy-6);
rectangle(_GBORDER,x+7,y+7,xx-7,yy-7);
setcolor(3);
moveto(x+6,y+6);
lineto(x+6,yy-6);
lineto(xx-6,yy-6);
moveto(x+7,y+6);
lineto(x+7,yy-7);
lineto(xx-6,yy-7);
setcolor(6);
rectangle(_GFillInterior,x+8,y+8,xx-7,yy-7);
}
int CubMenu1(int x,int y,int xx,int yy)
{
    setcolor(15);
    rectangle(_GBORDER,x,y,xx,yy);
    rectangle(_GBORDER,x+1,y+1,xx-1,yy-1);
    rectangle(_GBORDER,x+2,y+2,xx-2,yy-2);
    rectangle(_GBORDER,x+3,y+3,xx-3,yy-3);
    setcolor(8);
    moveto(x,y);
    lineto(x,yy);
    lineto(xx,yy);
    moveto(x+1,y+1);
    lineto(x+1,yy-1);
    lineto(xx-1,yy-1);
    moveto(x+2,y+2);
    lineto(x+2,yy-2);
    lineto(xx-2,yy-2);
    moveto(x+3,y+3);
    lineto(x+3,yy-3);
    lineto(xx-3,yy-3);
    setcolor(7);
    moveto(x+1,yy-1);
    lineto(x+4,yy-4);
    setcolor(15);
    moveto(xx,y);
    lineto(xx-4,y+4);
    setcolor(7);
    rectangle(_GFillInterior,x+4,y+4,xx-4,yy-4);
}
Bar3d(int x,int y,int xx,int yy,int depth)
{
    rectangle(_GFillInterior,x+2,y+2,xx-2,yy-2);
    setcolor(2);

```

```

rectangle(_GBORDER,x,y,xx,yy);
moveto(x,y);
lineto(x+depth,y-depth);
lineto(xx+depth,y-depth);
lineto(xx,y);
moveto(xx,yy);
lineto(xx+depth,yy-depth);
lineto(xx+depth,y-depth);
setcolor(7);
floodfill(x+depth/2,y-4,2);
setcolor(8);
floodfill(xx+depth/2,y+4,2);
}
Shadow(int x,int y,int xx,int yy)
{
    setcolor(7);
    rectangle(_GFillInterior,x+8,y+8,xx,yy);
    setcolor(8);
    rectangle(_GFillInterior,x,y,xx-8,yy-8);
    setcolor(1);
    rectangle(_GBORDER,x+2,y+2,xx-10,yy-10);
}

```

(本文收稿日期:1995.1.12)

编辑部致作者

来稿注意事项

(一)文字精炼、论点明确、数据可靠。

(二)正文一般不要超过八千字。用打印机打出或用原稿纸书写。程序一定要用打印机打出,墨色要足够深。程序要说明在什么条件下运行通过。

(三)论文要求包含内容提要、关键词,并附参考文献。同时附文章名的英译和作者名的汉语拼音。

(四)插图要求紧凑,用黑墨水画或用打印机打出,图中注字一般用小五号字,用激光机打的字剪贴(复盖在24点阵字上),条件许可者请用激光机出整图。插图的宽度一般控制在8cm或17cm之内。

(五)一般先不要寄软盘来,听候回复。在2个月不见回复的可再来函查询或按不录用考虑自行处理,请勿一稿多投。

(六)条件所限,来稿一律不退。确要退稿的请先付足邮资。

彩色立体窗口中文下拉式菜单编制技巧和示例

江西拖拉机发动机厂 黄焕如

内容提要:本文详细介绍了彩色立体中文下拉式菜单的编制方法和技巧,所列举的示例程序结构严谨、界面漂亮、操作方便,很容易扩充和修改。

关键字:C 语言,立体,下拉式菜单、窗口

众所周知,菜单在应用程序中的作用是很重要的。一个高质量的菜单不仅能为屏幕画面增添美观,还能让使用者觉得舒心 and 方便,防止了一些不必要的误操作,提高了工作效率,并能真正起到为应用系统锦上添花的作用。

彩色立体窗口中文下拉式菜单的应用特别广泛,其关键技术在于颜色的配制、立体窗口的实现、在西文或中文系统下调用汉字和下拉式菜单等一些问题。

下拉式菜单是典型的窗口菜单,自上而下向屏幕弹出一个个矩形块,矩形块内含有子菜单,子菜单又可根据需要分成多级菜单,逐级调用。下拉式菜单的设计关键是屏幕区域的保存和恢复问题,通常 C 语言提供的 `gettext()` 和 `puttext()` 函数以及其他有关窗口的函数在汉字操作系统下基本无效,而仅仅使用能适用于汉字系统的 `printf()`、`puts()`、`scanf()`、`gets()` 等标准输入输出函数制作高质量的菜单是很困难的。同时,汉字操作系统下显示的汉字仅仅为 16 点阵,色彩和显示方式比较单一,即使是 16 点阵汉字也只能在规定行列(例如 25 行,80 列)显示,例如需要在第 5、6 行之间显示汉字通常就不能实现,因此,采用直接读取汉字库中点阵数据的方法较好,并能够实现汉字的任意放大、各种颜色、不同字体、旋转方向等技术要求。利用光标键移动色棒,色棒的产生和消失采用图形屏幕操作函数 `getimage()` 和 `putimage()`,值得注意的是, `turbo c` 没有提供汉字屏幕窗口的保存和恢复功能,一般可通过类似于西文保存和恢复屏幕窗口的方法,通过 `malloc()` 函数分配内存,虽然其内存占用较西文几乎大一倍,但对于汉字不多的菜单其调用速度仍然较快并且使用很方便。

用 C 语言来实现汉字彩色立体窗口,可利用 `line()` 和 `setcolor()` 函数简单地编制一个新的函数 `windows()`,产生 45 度角平射时四边凸起的立体窗口,配合相应的色彩,就能达到很明显的立体效

果。

以下示例程序在屏幕上用淡洋红(LIGHT-MAGENTA)作衬底,主菜单在屏幕顶用青底色(CYAN)显示,用白色左上边框、黑色右下边框产生立体窗口,二级子菜单在相应的行位置显示。屏幕中间分左右两个大窗口,下部为信息提示窗口,均为淡灰色作底色,该程序在 `turbo c2.0` 下通过。

```
#include "stdio.h"
#include "graphics.h"
#include "stdlib.h"
#include "conio.h"
#include "bios.h"
#include "dos.h"
#include "process.h"
#include "alloc.h"
#include "fcntl.h"
```

```
int puthz16(int,int,int,int,char *);
int handle,test1,test2,test3,test4,test5,test6;
void windows();
```

```
main()
{
int gd=VGA,gm=VGAHI; int key,kkey,i;
initgraph(&gd,&gm,"");
cleardevice();
mainmenu();
closegraph(); exit(0);
}
```

```
int mainmenu()
{
/* 定义主菜单内容 */
unsigned char *{[]={"闭锁系统",
"仿真系统",
"信息系统",
"运行方式",
"自检验票",
"用户指南",
"退出系统"};
```



```

int i,x,key1=0,key2,test,size;
char c; void * buffer;
handle=open("hzk16",O_RDONLY|O_BINARY); /
* 打开汉字库 */
if(handle== -1)
{ cputs("Error on open hzk16");
  getch(); closegraph(); exit(1); }
i=0;
setfillstyle(SOLID_FILL,CYAN);
bar(0,6,getmaxx()-8,34);
setfillstyle(SOLID_FILL,LIGHTMAGENTA);
bar(0,40,getmaxx()-8,getmaxy());
setfillstyle(SOLID_FILL,LIGHTGRAY);
bar(8,50,444,420); windows(8,50,444,420);
setfillstyle(SOLID_FILL,LIGHTGRAY);
bar(470,50,620,420); windows(470,50,620,420);
setfillstyle(SOLID_FILL,LIGHTGRAY);
bar(8,434,620,470); windows(8,434,620,470);
setfillstyle(SOLID_FILL,CYAN);
bar(474,440,612,464); windows(474,440,612,464);

while(i<7) { windows(2+90*i,12,90*i+84,30);
  puthz16(12+90*i,14,0,BLUE,f[i]); i++; }

size=imagesize(3,12,85-2,30);
buffer=malloc(size);
getimage(3,12,85-2,30,buffer);
putimage(3,12,buffer,NOT_PUT);

x=3;
while(1)
{
  while(key1!=13) /* 按回车键退出循环 */
  {
    while(bioskey(1)==0); /* 等待键盘输入 */
    key1=key2=bioskey(0); /* 取键盘输入码 */
    key1=key1&0xff; /* 只取 ASCII 码 */
    key2=key2&0xff? 0:key2>>8; /* 只取扩充码 */
    if(key2==75||key2==77) /* 如果为左,右箭头键 */
    {
      putimage(x,12,buffer,COPY_PUT);
      /* 恢复原来信息 */
      if(key2==77) x=x>=543? 3:x+90; /* 右箭头处理 */
      if(key2==75) x=x<=3? 543:x-90; /* 左箭头处理 */
      getimage(x,12,x+83-2,30,buffer);
      /* 保存新位置色棒大小内容 */
      putimage(x,12,buffer,NOT_PUT);
      /* 反象释放产生色棒 */
    }
  }
  test=(x-3)/90+1;
  switch(test) /* 根据不同选择分项处理 */
  {

```

```

    case 1:
      menu1(); break;
    .....
    case 6:
      menu6(); break;
    default:
      closegraph(); exit(0);
  }
  key1=0;
}
free(buffer); /* 释放内存 */
}

int menu1()
{ }
.....
int menu6() /* 第 6 子菜单 */
{
  unsigned char *f[]={"系统简介",
    "使用手册",
    "维护手册"};
  int i,y6,ke61,ke62,si60,si61;
  void * bu60,* bu61;
  si60=imagesize(450,50,536,140);
  bu60=malloc(si60);
  getimage(450,50,536,140,bu60);
  setfillstyle (SOLID_FILL, CYAN); bar (450, 50, 536,
    140);
  windows (456,57,530,75); puthz16(461,58,0,BLUE,
    f[0]);
  windows (456, 85, 530, 103); puthz16 (461, 86, 0,
    BLUE,f[1]);
  windows (456, 113, 530, 131); puthz16 (461, 114, 0,
    BLUE,f[2]);

  si61=imagesize(456,57,530,74);
  bu61=malloc(si61);
  getimage(456,57,530,74,bu61);
  putimage(456,57,bu61,NOT_PUT);

  y6=57,ke61=0;
  while(ke61!=13&&ke61!=27)
    /* 按回车键或 ESC 键退出循环 */
  {
    while(bioskey(1)==0); /* 等待键盘输入 */
    ke61=ke62=bioskey(0); /* 取键盘输入码 */
    ke61=ke61&0xff; /* 只取 ASCII 码 */
    ke62=ke62&0xff? 0:ke62>>8; /* 只取扩充码 */
    if(ke62==72||ke62==80) /* 如果为上,下箭头键 */
    {
      putimage(456,y6,bu61,COPY_PUT);
      /* 恢复原来信息 */
      if(ke62==72&&(y6==85||y6==113)) y6=y6-
        28; /* 上箭头处理 */
      else {if(ke62==72&&y6==57) y6=y6+28*2;
        if(ke62==80&&(y6==57||y6==85)) y6=y6+
          28; /* 下箭头处理 */
      }
    }
  }
}

```

用C语言实现图形方式下的花样清屏

电子工业部第四十五研究所(平凉 744000) 丁永峰

摘 要: 本文介绍了在 EGA/VGA 图形方式下,采用了多种新颖别致的清屏方法,即自上至下的瀑布式和从中间向两边拉的拉幕式、两边向中间拉的闭幕式、从左至右或从右至左式、旋转方式等。此种方法用 Turbo c 2.0 实现,并给出了详细的程序清单。

关键词: 清屏 图形模式 图形系统的初始化 图形适配器

一. 引言

图形显示器以其丰富多彩的颜色,在图像处理、工业控制、信息管理系统等领域中得到了广泛的应用。在图形方式下,屏幕显示的清除在很多应用软件中是不可缺少的。然而,清屏的方法很多,使用 DOS 的清屏命令和 C 语言提供的清屏函数,大家都知道其效果只是单一的清除屏幕显示,谈不上什么艺术性,若想设计丰富多彩的用户界面,自己编写清屏程序是必要的。在这方面许多报刊对文本方式下的清屏方法谈论得较多,而图形方式下的介绍甚少。有鉴于此,笔者把自己参与一项国家“八五”科研攻关项目工作中,其中计算机实时控制的用户界面所采用的图形模式下的花样清屏作一介绍。多花样的清屏,可使用户界面优美,给操作者以赏心悦目的视觉感受。

Turbo c 2.0 提供了许多功能强大的图形函数,用 C 函数来完成图形屏幕的清除是比较方便的。本文介绍的方法,直接用 C 函数编写。

二. 关于图形显示器和图形模式

常用显示器的性能见表 1。

表 1 常用显示器的性能

型 号	年代	分辨率	颜色
CGA(Color Graphics Adapter)	1981	320 * 200 640 * 200	4 2
Herclous(herclous monochrome graphics adapter)	1982	720 * 348	1
EGA(Enhanced Graphics Adapter)	1984	640 * 200 640 * 350	16 16
VGA(Video Graphics Array)	1987	640 * 200 640 * 480	16 16
TVGA(Enhancement-mode Video Graphics Array)	1989	640 * 400 640 * 480 800 * 600 1024 * 768	256 256 256 256

在图形模式,整个屏幕按显示器的分辨率分成点阵,左上角为(0,0),X轴从左到右(0~639),Y轴从上到下(0~479;对VGA)。图形模式中,也可以定义一个长方形的区域,称之为图形窗口(viewport),当图形程序作图时,只在 viewport 内有效,viewport 之外的屏幕是不可接触的。可以用 setviewport() 函数来定义 viewport 的位置和大小。

三. 程序中使用有关作图函数介绍

① void far detectgraph(int * gdriver, int * gmode)

其中 gdriver 和 gmode 分别表示图形驱动器和图形模式,有关常数在表 2 和表 3 中介绍,定义在 graphics.h 中,调用这个函数后,有关图形显示器的驱动器和图形模式的情况就从这两个参数返回给调用函数。

表 2 有关图形驱动器符号常数以及数值规定

符号常数	数值	意 义
DETECT	0	根据硬件测试结果,自动装入相应驱动程序
CGA	1	CGA 显示器
MCGA	2	Multi color Graphics Array 显示器
EGA	3	EGA 显示器
EGA64	4	EGA 64 显示器
EGAMONO	5	EGA 单色显示器
IBM8514	6	IBM 8514 显示器
HERCMONO	7	Herclous 显示器
ATT400	8	AT&T 400 行图形显示器
VGA	9	VGA 显示器
PC3270	10	PC 3270 显示器

表3 有关图模式的符号常数

图形驱动	图形模式	数值	列 * 行	色调
CGA	CGAC0	0	320 * 200	C0
	CGAC1	1	320 * 200	C1
	CGAC2	2	320 * 200	C2
	CGAC3	3	320 * 200	C3
	CGAHI	4	640 * 200	2 色
MCGA	MCGAC0	0	320 * 200	C0
	MCGAC1	1	320 * 200	C1
	MCGAC2	2	320 * 200	C2
	MCGA3	3	320 * 200	C3
	MCGAMED	4	640 * 200	2 色
	MCGAHI	5	640 * 480	2 色
EGA	EGAL0	0	640 * 200	16 色
	EGAHI	1	640 * 350	16 色
EGA64	EGA64L0	0	640 * 200	16 色
	EGA64HI	1	640 * 350	4 色
EGAMON	EGAMONHI	0	640 * 350	2 色
HERC	HERC-MONOH	0	720 * 348	2 色
ATT400	ATT400C0	0	320 * 200	C0
	ATT400C1	1	320 * 200	C1
	ATT400C2	2	320 * 200	C2
	ATT400C3	3	320 * 200	C3
	ATT400MED	4	640 * 200	2 色
	ATT400HI	5	640 * 400	2 色
VGA	VGAL0		640 * 200	16 色
	VGAMED1	1	640 * 350	16 色
	VGAHI2	2	640 * 480	16 色
PC3270	PC3270HI	0	720 * 350	2 色
IBM8514	IBM8514LO	0	640 * 480	256 色
	IBM8514HI	1	1024 * 768	256 色
TVGA	TVGA8900	5Dh	640 * 400	256 色
		5Dh	640 * 480	256 色
		5Eh	800 * 600	256 色
		62h	1024 * 768	256 色
		6Ch	640 * 400	16M
	TVGA9200	70/71h	521 * 480	32/64K
		74/75h	640 * 480	32/64K
		76/77h	800 * 600	32/64k

② void far initgraph (int * gdrive, int * gmode, char * path)

其中 gdriver 和 gmode 和①介绍相同, path 是指图形驱动程序存放在那个目录中, 此函数也称图形系统的初始化函数。因为图形显示器种内很多, 控制方式各异, 要显示图形, 必须首先装入相应的图形驱动程序, 其次每一种图形显示器, 又具有几种不同的图形显示模式。要显示图形前必须决定选用哪一种显示模式, 这就是图形系统初始化工作。

③ void far setbkcolor(int color)

设置图形的背景颜色

④ void far setcolor(int color)

设置现行作图颜色, setcolor 是规定画笔的颜色, 背景色就好比纸的颜色。

⑤ void far bar(int x1, int y1, int x2, int y2)

先画一长方形, 再填充。其中 (x1, y1) 是起始坐标点, (x2, y2) 是终点坐标点。

⑥ line(int x1, int y1, int x2, int y2)

画一条直线从 (x1, y1) 到 (x2, y2), 这个函数是清屏操作的关键函数。因为大家都知到, 点的运动可形成线, 线的运动可形成面, 正是通过这一原理使得线在屏幕上移动, 从而可达到清屏的目的。

程序清单如下:

```

/* * filename: clscreen.c * */
#include <graphics.h>
int i;
main()
{
    int gdriver, gmode;
    detectgraph(&gdriver, &gmode);
    if(gdriver < 0)
    {
        printf("There is not graphics display ! \n");
        exit(1);
    }
    printf("Detect Graphics driver is # %d, mode is # %d\n", gdriver, gmode);
    printf("Press key continue\n");
    getch();
    initgraph(&gdriver, &gmode, "c:\\tc");
    clsb();
    dispchar();
    initgraph(&gdriver, &gmode, "c:\\tc");
    clsd();    clsa();
    clsc();    clse();
    clsf();    clsg();
    clsh();    clsj();
    clsl();    clsr();
    clsu();    clsv();
    clsw();
    setcolor(15);
    outtextxy(200, 450, "Press any key to exit. ....");
    getch();
    closegraph();
}

int dispchar(void)
{
    setbkcolor(15); setcolor(9);
    setlinestyle(0, 0, 3);
    setfillstyle(9, 14);
    rectangle(4, 4, 635, 475);
    line(4, 10, 635, 10);
    setfillstyle(SOLID_FILL, 14);
}

```

```

bar3d(150,440,490,460,0,0);
setcolor(2);
outtextxy(200,450,"Press any key to continue....");
setlinestyle(4,0,3);
settextstyle(1,0,0);
setusercharsize(2,2,7,5);
outtextxy(60,120,"MODEL BG -- 102 WAFER
STEPPER");
setusercharsize(2,2,2,2);

/* * 由中间向上下清 * */
clsa()
{
    setbkcolor(2);
    setcolor(5);
    for (i=239; i>=0; i--)
    {
        line(0, i, 639, i);
        delay(5);
        line(0, 479-i, 639, 479-i);
        delay(5);
    }
}

/* * 由上下向中间清 * */
clsc()
{
    setbkcolor(3);
    setcolor(14);
    for (i=0; i<=239; i++)
    {
        line(0, i, 639, i);
        delay(5);
        line(0, 479-i, 639, 479-i);
        delay(5);
    }
}

/* * 由中间向左右清 * */
clsb()
{
    setbkcolor(4);
    setcolor(15);
    for (i=319; i>=0; i--)
    {
        line(i, 0, i, 479);
        delay(5);
        line(639-i, 0, 639-i, 479);
        delay(5);
    }
}

/* * 由左右向中间清 * */
clsd()
{
    setbkcolor(15);
    setcolor(4);
    for (i=0; i<=319; i++)
    {
        line(i, 0, i, 479);
        delay(5);
        line(639-i, 0, 639-i, 479);
        delay(5);
    }
}

/* * 由上向下清 * */
clse()
{
    setbkcolor(3);
    setcolor(6);
    for (i=0; i<=479; i++)
    {
        line(0, i, 639, i);
        delay(5);
    }
}

/* * 由下向上清 * */
clsg()
{
    setbkcolor(14);
    setcolor(5);
    for (i=479; i>=0; i--)
    {
        line(0, i, 639, i);
        delay(5);
    }
}

/* * 由左向右清 * */
clsf()
{
    setbkcolor(5);
    setcolor(3);
    for (i=0; i<=639; i++)
    {
        line(i, 0, i, 479);
        delay(5);
    }
}

/* * 由右向左清 * */
clsh()
{
    setbkcolor(4);
    setcolor(15);
    for (i=639; i>=0; i--)
    {
        line(i, 0, i, 479);
        delay(5);
    }
}

outtextxy(76, 290," THE 45TH RESEARCH INSTI-
TUTE");
setusercharsize(2,3,2,2);
outtextxy(140,330,"MINISTRY OF ELECTRONIC IN-
DUSTRY");
setusercharsize(2,2,1,2);
outtextxy(185,390,"SEPTEMBER 1994");
getch();
}

delay(5);
line(639-i, 0, 639-i, 479);
delay(5);
}

/* * 清除四个角 * */
clsj()
{
    setbkcolor(5);
    setcolor(9);
    for (i=0; i<=239; i++)
    {
        line(0, 239-i, 319, 239-i);
        line(319-i, 0, 319-i, 239);
        delay(5);
        line(319, 239-i, 639, 239-i);
        line(319+i, 0, 319+i, 239);
        line(0, 239+i, 319, 239+i);
        line(319-i, 0, 319-i, 479);
        delay(5);
        line(319, 239+i, 639, 239+i);
        line(319+i, 0, 319+i, 479);
        delay(5);
    }
}

/* * 顺时针旋转 180 度清屏 * */
clsl()
{
    setbkcolor(4);
    setcolor(15);
    for (i=1118; i>=0; i--)
    {
        if (i<=319)
            line(319-i, 0, 319+i, 479);
        else if (i>319 && i<=798)
            line(0, i-319, 639, 798-i);
        if (i>798)
            line(i-798, 479, 1437-i, 0);
    }
}

/* * 逆时针旋转 360 度清屏 * */
clsv()
{
    setbkcolor(4);
    setcolor(15);
    for (i=0; i<=2236; i++)
    {
        if (i<=319)
            line(319-i, 0, 319, 239);
    }
}

```



```

else if(i>319 && i<=798)
    line(0, i-319, 319, 239);
if(i>798 && i<=1437)
    line(i-798, 479, 319, 239);
else if(i>1437 && i<=1916)
    line(640, 1916-i, 319, 239);
if(i>1916 && i<=2236)
    line(2555-i, 0, 319, 239);
}
}

/* * 由左上角和右下角向中间清屏 * */
clrscr()
{
    setbkcolor(5);
    setcolor(14);
    for (i=639; i>=0; i--)
    {
        line(0, 479-i, 639-i, 0);
        delay(5);
        line(i, 479, 639, i);
    }
}

/* * 由中间向四周清屏幕 * */
clsw()
{
    setbkcolor(12);
    setcolor(15);
    for (i=0; i<=319; i++)
    {
        bar(310-i, 250-i, 328+i, 248+i);
        delay(5);
    }
}

/* * 由左上角向右下角清屏 * */
clsu()
{

```

四. 结束语

上述程序是在 AST 386、联想 386、COMPAQ 386/486 计算机

VGA 适配器上运行通过。假如使用的不是 VGA 显示器,采用此程序也无妨,只要根据测试函数测得图形驱动器的值,再根据表 2 和表 3 来修改循环次数 i 值,也可以达到同样的效果。经过以上的努力,我们完全实现了图形方式下的花样清屏,这给枯燥的屏幕和用户界面带来一份绚丽的色彩。本文的抛砖引玉至此结束,祝读者朋友更上一层楼。

参考文献

- [1]徐金悟、刘冶钢、张思宇编译,《Turbo c 使用大全(V1.5 ~ V2.0)》,北京科海培训中心,1989 年 12 月
- [2]叶欣 编译,《Turbo c V2.0 参考手册》,北京科学院希望高级电脑技术公司,1990 年 5 月

(本文收稿日期:1994.12.18)

(上接 25 页)

```

else {if(ke62==80&&y6==113) y6=y6-28*2;}
getimage(456,y6,530,y6+19,bu61);
/* 保存新位置色棒大小内容 */
putimage(456,y6,bu61,NOT_PUT);
/* 反象释放产生色棒 */
}
free(bu61); /* 释放内存 */
if(ke61==13) test6=(y6-57)/28+1;
if(ke61==27) test6=0;
if(test6==1) /* 分项处理 */
if(test6==2)
if(test6==3)
putimage(450,50,bu60,COPY_PUT);
free(bu60);
}

int getbit(unsigned char c,int n)
{ return((c>>n)&1); }
int puthz16(int x,int y,int z,int color,char *p)
{ /* 显示 16 点阵字 */
    unsigned int i,c1,c2,f=0;
    int i1,i2,i3,rec;
    long l; char by[32];
    while((i=*p++)!=0)
    { if(i>0xa1)
        if(f==0) { c1=(i-0xa1)&0x07f; f=1; }
        else
        { c2=(i-0xa1)&0x07f; f=0;
            rec=c1*94+c2;

```

```

l=rec*32L;
lseek(handle,l,SEEK_SET);
read(handle,by,32);
for(i1=0;i1<16;i1++)
for(i2=0;i2<2;i2++)
for(i3=0;i3<8;i3++)
    if(getbit(by[i1*2+i2],7-i3))
        putpixel(x+i2*8+i3,y+i1,color);
x=x+16+z; }
}
return(x);
}

```

```

void windows(x1,y1,x2,y2) /* 立体窗口 */
int x1,x2,y1,y2;
{ int i;
    setcolor(WHITE);
    for(i=1;i<3;i++) line(x1-i,y1-i,x2+i,y1-i);
    for(i=1;i<3;i++) line(x1-i,y1-i,x1-i,y2+i);
    setcolor(DARKGRAY);
    for(i=1;i<3;i++) line(x2+i,y1-i,x2+i,y2+i);
    for(i=1;i<3;i++) line(x1-i,y2+i,x2+i,y2+i);
}

```

参考文献:

- [1]袁津生,《C 语言实用技巧》,北京师范大学出版社,1993 年 9 月
- [2]庄粤盛,《TURBO C 绘图程序设计实例》,学苑出版社,1993 年 10 月

(本文收稿日期:1994.12.10)

单片机和单板机加密实用技术

郑州解放军电子技术学院 王成义

摘要: 本文介绍了单片机和单板机在信息传输和存储中的安全保密,利用单片机内部封锁电路对存储信息进行加密,以达到保密的目的。具体介绍了单片机加密的方法,并给出了实用程序。

关键词: 单片机 信息 保密

一、引言

在当今高新科学技术飞速发展的时代,信息已成为推动社会向前发展的巨大资源。微电子技术的不断发展,使处理信息的技术手段也在不断地提高,更多的业务工作变得越来越依赖电子信息处理了。而且由于信息资源有别于其它资源,它的脆弱性更加明显。信息资源不同于一般的资源,在于它可以同时被许多人占有而共同利用。因此,在信息处理中,信息的安全和保密是至关重要的。在信息传输和存储中安全保密有三个主要方面,一是安全性,二是保密性,三是完整性。所谓信息的保密性,是将信息进行适当的变换,使之成为“面目全非”的密码信息,然后将这些密码信息进行传输和存储,以达到局外人无法了解信息内容的目的。而信息的完整性保护是指不容许对传输或存储中的信息进行篡改、增加、删除和伪造等,并防止用过期的信息冒充当前的信息。随着单片微机不断提高和发展,它在自动控制、测量、智能化仪器仪表等领域得到广泛应用,因此,单片机和单板机的防复制、防伪造以及应用程序的信息保密已成为一个极其重要的问题。

二、具有程序保密措施的单片机

在 MCS-51 单片机系列中,新近开发研制了具有程序存储器封锁措施的器件,主要有 8751H、8751BH、87C51H 和 8752BH 等。在这些单片机中设置了程序存储器封锁电路,该线路阻止了非法破译者读取被保护的软件程序,保护了软件版权不受侵犯。

8751H 单片机片内有 4KB EPROM,是用新技术制造的,功能强。其内部具有一个封锁位,当将该封锁位置位后,8751H 单片机内的程序存储器即对外部关闭,任何的外部手段均无法访问。此时,对于该器件既不能进一步编程,也不能执行外部程序存储器的指令,但仍然可以执行内部存储器的指

令。若想恢复器件的原有全部功能,只有将 EPROM 擦除后才能撤销封锁状态,此时 EPROM 才能再次进行编程。

8751BH、87C51H 等 MCS-51 单片机器件内部具有更强的保密部分。它内部有两个封锁位,同时还有 32 个字节的加密阵列,具有进一步进行加密的作用。通过对这 32 个字节的加密阵列编程,使其能在对 EPROM 校验时把固化程序变换为密码。在加密检验的过程中,每访问一个程序字节,都从加密阵列中选出一个字节和所访问的字节进行异或非逻辑运算,产生一个加密检验字节。加密阵列中的各字节按顺序投入运行。因此,要将检验结果还原为程序代码,必须知道这 32 个字节的正确顺序。

三、利用单片机程序初始化进行保密

每一个单片机应用系统都要对整个系统进行初始化。一般情况下,系统初始化的主要工作就是对有关的端口设置初始状态。对相应工作单元进行清零;设置中断控制字和各中断的优先级;设置定时器工作方式;设置堆栈指针以及设置数据等等。通过初始化程序,将一些初始数据送到相应的工作单元和功能寄存器以及端口中。因此,在整个系统运行之前,通过初始化加密处理程序,将单片机内部 RAM 中有关工作单元清零或设置数据,并且把对功能寄存器以及端口初始化所需的参数也存入单片机内部 RAM 中,利用单片机具有掉电保护功能,在系统掉电后内部 RAM 的内容保持不变的特点,在主程序中,把断电后保存在单片机内部 RAM 中的初始化参数传送到相应的功能寄存器以及端口中,从而达到程序初始化保密的效果。

现将一个单片机应用系统的初始化工作中,对单片机内部 RAM 的工作单元 15H-22H 分别设置成 22H、31H、40H、49H、58H、67H、76H、85H,工作单元 3AH-3FH 清零,堆栈指针设置成 20H,定

时器模式控制寄存器的内容设置为 05H, 计数寄存器 TL0 和 TL1 的内容, 分别设置为 FFH 和 1AH, 允许中断寄存器的内容设置为 87H。通过以下程序段来完成对单片机内部 RAM 的相应的工作单元清零或者设置数据, 并且使用部分内部 RAM 的存储单元保存相应的功能寄存器和端口的初始化参数。

```

ORG    0000H
SJMP   SET_PW
ORG    0003H
SJMP   PROTE
ORG    0040H
SET_PW: MOV    A, #00H
        ;将内部 RAM3AH-3FH 清零
        MOV    R0, #3AH
        MOV    R1, #06H
MEM_ZERO: ZE-MOV @R0, A
        INC    R0
        DJNC   R1, MEM_ZERO
        SETB   EX0
        ;允许外部 0 号中断
        MOV    15H, #22H
        ;对 15H-22H 分别设置数据
        MOV    16H, #31H
        MOV    17H, #40H
        MOV    18H, #49H
        MOV    19H, #58H
        MOV    20H, #67H
        MOV    21H, #76H
        MOV    22H, #85H
        MOV    21H, #20H
        ;保存堆栈指针
        MOV    22H, #05H
        ;保存定时器模式寄存器内容
        MOV    23H, #0FFH
        ;保存计数寄存器低位
        MOV    24H, #1AH
        ;保存计数寄存器高位
        MOV    25H, #087H
        ;保存允许中断寄存器内容
WAIT:    NOP
        SJMP   WAIT
PROTE:   ORL    PCON, #02H
        ;设置为掉电工作方式

```

当以上加密程序在单片机系统上运行并掉电后, 系统初始化所需相应的初始参数和工作单元即设置完毕。这样只要在主程序中将保存在单片机内部 RAM 中的相应初始参数取出, 同时送到相应的功能寄存器中, 整个单片机系统的初始化及保密工作也就全部完成了。此时, 系统就能正常运行。而对破译者来说, 虽然经过系统初始化加密处理的应用程序可以轻易地被非法复制, 但复制的应用程序却无法在仿制系统上正常运行。

四、利用单片机内设置保密字进行保密

对于一些具有掉电保护功能的单片机, 我们可以通过在单片机内部的 RAM 中设置保密字的方法来实现对应用程序的加密保护。针对单片机掉电后, 单片机内部 RAM 的内容将保持不变的特点, 结合自己的应用程序, 在单片机内部 RAM 中事先设置好保密字。当系统运行时, 应用程序通过经常测试保密字的正确性来控制应用程序的流向, 实现加密保护。假设单片机内部 RAM 中的 30H 和 31H 存储单元用作保密的存储单元, 两个保密字存储单元的内容分别为 15H 和 30H (可以根据自己的需要而设置保密字的个数和内容)。在系统运行固化在 EPROM 中的主程序之前, 必须事先在单片机内部 RAM 中设置好保密字。下面程序段就是用来设置系统保密字。

```

ORG    0000H
SJMP   SET_PW
        ;转向设置保密字
ORG    0003H
SJMP   PROTE
        ;转向掉电保护中断服务程序入口
ORG    0040H
SET_PW: MOV    30H, #15H
        ;设置保密字
        MOV    31H, #30H
        MOV    IP, #01H
        ;设置外部 0 号中断(断电保护中
        ;断)为高优先级中断
        SETB   EX0
        ;允许外部 0 号中断
WAIT:    NOP
        ;等待掉电中断
        SJMP   WAIT
PROTE:   ORL    PCON, #02H
        ;设置成掉电工作方式
END

```

设置保密字程序中标号 PROTE 为掉电保护中断服务程序入口, 先把设置保密字程序写入 EPROM 中, 然后插入单片机系统, 使系统运行后再掉电, 此时保密字就设置好了。然后拔掉设置保密字的 EPROM, 再插入应用程序的 EPROM, 可以在主程序中加入一系列的运算和研判程序来控制程序的流向。

参考文献:

- [1] 陈爱民等著,《计算机的安全与保密》, 电子工业出版社, 1992 年
- [2] 杨迈等著,《软件加密/解密及反跟踪实用技术》, 西安电子科技大学出版社, 1991 年

(本文收稿日期: 1994. 12. 14)

如何在自制软件中加入五笔输入法

广西柳州市代豪电脑公司 麦智祥

内容提要:本文论述通过对金山排版系统中五笔输入法文件(WBX.COM)的分析,并从中提取五笔字编码文件,由该编码文件中逆推出某输入编码所对应的汉字。

关键词:自制软件 五笔输入法

在很多自己开发的软件中,都不可避免地涉及到汉字输入、输出的问题,因此,很多软件都需要依赖运行汉字系统来解决此问题,但如此又会引起系统过于庞大。那么,如何在自行开发的软件中实现汉字的输入输出呢?

汉字的输出较容易实现,主要方法就是从字库中将所需汉字点阵读出并显示在屏幕上,这已有很多文章介绍,我们关心的主要是如何实现汉字的输入,尤其是用五笔字型输入法来输入汉字。下面,本文将详细地说明如何实现这一点。

第一步:生成五笔编码表文件

从金山排版系统 5.1 版本中的 WBX.COM 文件中提取五笔编码文件。在该文件中存放着五笔编码表,其偏移位置为 0x6AC,每个汉字的编码占用 3 个字节。国标一、二级汉字共有 6763 个,所以,编码表长度应该为 $6763 \times 3 = 20289(0x4F41)$ 。

用 DEBUG 将编码表以 WBM.DAT 文件存盘,方法如下:

```
debug wbx.com
-N WBM.DAT
-RCX
BA4F
:4F41
-W 7AC
-Q
```

第二步:分析五笔编码表格式

编码表中按区位码顺序依次存放着国标一、二级汉字共 6763 个,每个汉字用 3 个字节 24 位表示,其中 b24—20 为第一字根, b19—17 再加上 b14—13 为第二字根, b12—9 加上 b6 为第三字根, b5—1 为末字根, b16、15 为简码代码, b7—8 为重码代码。

例:“啊”字为编码表中首个汉字,其编码为 0x58,0x59,0x2B

58 E9 2B

01011000 11101001 00101011

按以上分析方法,不难得出其第一、二、三及末字根码为 11,2,19,11,所对应的字母 KBSK 即是其五笔字根码。

而当 b16 为 1 时表明该字是 2 级简码, b15 为 1 时表明该字是 3 级简码,据此,我们又可得知“啊”字既是 2 级简码又是 3 级简码。

第三步:编程实现五笔字型输入法

本文最后给出一个 C 语言例子,具体说明如何实现五笔输入法。

为减少频繁读磁盘,提高运行速度,我们将编码表读入内存。当使用者输入五笔编码后,调用转换子程序。

子程序首先将所输入编码与高频字(一级简码)比较,如非高频字,再将所输入编码与 6763 个汉字的五笔编码分别比较,并在比较过程中处理重码及学习键,最后根据情况返回空字符串(输入错码),单个汉字或一串汉字(重码)。

至此,我们已基本可以在自行开发的软件中加入五笔字型输入法了,但该方法仍有一点不足,就是运行速度稍慢,用以和商品化的汉字系统相比便知道,下一步的改进工作,就留给您来完成吧!

```
#include "stdio.h"
#include "alloc.h"
#include "string.h"
#include "fcntl.h"
#include "io.h"
void readfile();
char * wbscr, * wbpoint, * convwb();

char * convwb(char * inp)
{
```

```

char * nol[]={"工","了","以","在","有","地","一",
"上","不","是","中","国","同","民","为","这","我","的","要",
"和","产","发","人","经","主",""};

/*
如输入为错码, 返回 "",
否则返回单个或一串汉字.
*/
unsigned char a,b,c;
unsigned char hz[30];
unsigned char aa[10]=" ";
int ch,i,j,ji=4,end,len,dupp=0,d;
while(1)
{
if (strcmp(inp,"")==0) return "";
if (strlen(inp)==1) /* 处理是否为高频字 */
{
ji=( * inp&0x9f)-1;
sprintf(hz,"%s",nol[ji]);
return hz;
}
(void)strupr(inp); /* 转换为大写 */
wbpoint=wbscr;
for(i=0;i<=6763;i++)
{
end=0;
a= * wbpoint;
a=(a==255)? 26 : a;
wbpoint++;
b= * wbpoint;
b=(b==255)? 26 : b;
wbpoint++;
c= * wbpoint;
c=(c==255)? 26 : c;
wbpoint++;
* aa=a/8+64;
* (aa+1)=(a&7)*4+(b/16&3)+64;
* (aa+2)=(b&15)*2+(c/32&1)+64;
* (aa+3)=(c&31);
* (aa+3)=(( * (aa+3)<1|| * (aa+3)>25)? 32: * (aa+3)+64);
* (aa+4)='\0';
if ( * (aa+3)==' ')
* (aa+3)='\0';
ji=(b/64&1)? 3: 4; /* 此两句分析其简码 */
ji=(b/64&2)? 2: ji;
if ( strlen(inp)<=ji )
len=ji;
if ( strlen(inp)>ji )

```

```

len=strlen(inp);
for(j=0;j<len;j++)
if ( * (inp+j)!= * (aa+j) && * (inp+j)!=
'Z')
{
end=1;
break;
}
if (end==1) continue;
* (hz+2 * dupp)=i/94+176;
* (hz+2 * dupp+1)=(i%94)+161;
* (hz+2 * dupp+2)='\0';
dupp++;
}
if (dupp<1) return "";
return hz;
}
}
void readfile()
{
/* 将编码表读入内存 */
int han;
long off;
han=_open("wbm.dat",O_RDONLY);
off=lseek(han,0,SEEK_END);
lseek(han,0,SEEK_SET);
wbscr=(char *)malloc(off);
read(han,wbscr,off);
close(han);
wbpoint=wbscr;
}
void main()
{
char incode[6], * string;
readfile();
while(1)
{
printf("\nPlease Input Code:");
scanf("%s",&incode);
if (strcmpi(incode,"END")==0)
break;
string=convwb(incode);
if (strcmp(string,"")!=0)
printf("\n%s",string);
else printf("\nError");
}
free(wbscr);
}

```

(本文收稿日期:1995.1.18)

屏幕图象彩色硬复制的 C 语言编程方法

武汉[430073] 邓中明 刘光越

摘 要:本文介绍一种利用针式彩色打印机复制彩色屏幕图形的功能函数,该函数可为 Turbo C, Borland C++ 所直接调用。

关键词:屏幕图象, 彩色打印, Turbo C/C++

1. 引 言

C 语言以其简洁, 灵活, 高效的优点, 并具有彩色绘图等功能而倍受计算机用户及工程技术人员的青睐。采用 VGA 显示器具有高分辨率, 多颜色等优点, 就可制作精美, 多彩的图象。美中不足的是 C 语言系统一般不具备彩色屏幕图形复制功能。若用一般的方法调用 int 5h 中断, 只能在单色打印机上作彩色屏幕的黑色硬复制(Hard copy)。文章[1][2]中介绍了彩色屏幕的单色打印或彩色屏幕的多灰度打印。

随着彩色打印机的应用也愈来愈普及, 广大计算机用户希望把彩色屏幕复制下来。为此, 本文介绍了一种利用针式彩色打印机进行彩色打印方法, 并针对 CR3240 彩色打印机编制了一个屏幕图象的彩色硬复制通用函数。

2. 彩色硬复制原理

本文以应用最广泛的 VGA (TVGA) 640 × 480 分辨率, 16 色的模式为例, 说明彩色屏幕图象打印原理。

文[2]中已对 24 针打印机对屏幕单色硬复制原理作过介绍, 本文的彩色打印基本方法也基于[2]的原理, 再利用彩打的不换行重复打印等功能, 将不同彩色屏幕点阵循环识别时将屏幕上 24 行, 按屏幕上存在的彩色分别进行图象的识别, 再重复打印。

① 设置打印彩色原理

[3]在计算机图形学中有两种重要的原色配色系统, 即红, 绿, 蓝(RGB)加色系统和青, 品红, 黄(CMY)减色系统(见图 1), 两种系统中的颜色互为补色, 对于彩色色带常采用 CMY 减色系统, 而对于彩色 CRT 显示器常采用 RGB 加色系统。两个物色系统的混合与一光色的原色相同。

彩色打印色带一般只有四种基本色, 即⊙黑色 ①品红 ②青 ④黄, ③

蓝, ⑤红, ⑥绿。仔细观察彩打的打印过程就会发现, 彩打的基本打印色彩③, ⑤, ⑥号色是利用①, ②, ④的叠加打印而成(即⑥=②+④, ⑤=①+④, ③=①+②)。本文的彩打方法是利用该原理, 即利用物色三原色的合成可打印出多彩的图象。

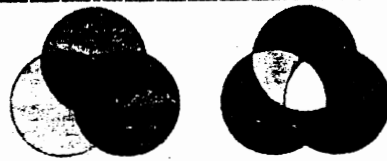
② 光色三原色和物色三原色的关系

VGA 显示器的 16 种(或 256 种)颜色是利用光色三原色的 R(红), G(绿), B(蓝)合成。而彩色

表 1

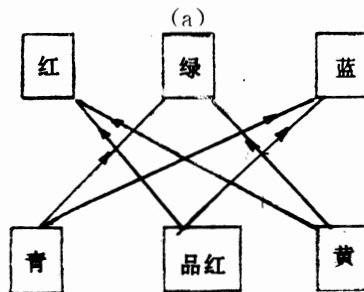
VGA 颜色代号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
颜色 名称	黑 色	蓝 色	绿 色	青 色	红 色	洋 红 色	棕 色	淡 灰	深 灰	淡 蓝	淡 绿	淡 青	淡 红	淡 洋 红	黄 色	白 色
彩色打 印合成	I	①	③	⑥	②	⑤	①	⑤	⑧	①	③	⑥	②	⑤	①	④
颜 色	II	⑥	②	⑥	②	⑤	①	④	⑥	②	⑥	②	⑥	②	⑥	⑥
颜 色	III	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙	⊙

注: ⊙ 等于 0



物色混色(减色系统)

光色混色(加色系统)



(b)

图 1

打印机色带的三原色是物色三原色(另加黑色),为把屏幕光色三原色合成的颜色用物色三原色叠加打印出来,必须先确定光色合成色与物色合成色之间的关系。见图(1)。利用图(1)的对应关系(即⑥=②+④)就可用彩打的七种基本打印颜色配置叠加打印出 VGA 光色合成的颜色,(见表 1)。

此外,还可充分利用 CR3240 彩打的重打(ESC+'E'),重复打印(ESC+'G'),高速打印(ESC+x1),高密打印(ESC+x0),高速高密打印(ESC+x2),消音打印(ESC+s1)等功能打印出彩色灰度效果。

3. CR3240 彩色打印程序

```
void rdscr0(int mode)
{ /* mode:32,33,38,39,40 打印图形模式 */
  unsigned char by[3][640];
  struct viewporttype v;
  int col[3][16]={ /* 16 色仿真颜色
    0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 */
    0,3,6,2,5,1,5,8,1,3,6, 2, 5, 1, 4, 1,
    0,2,6,2,5,1,4,0,6,2,0, 0, 0, 0, 0, 6,
    0,0,0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0};
  register int i,j,x,y,ic;
  int k,cor,ki,kg,bk,ik,kind,XM,YM0,YM1,kkk,xmax,
  ymax,cmax;
  long l; char c,ch;
  getviewsettings(&v); /* 保存原视见区 */
  xmax=getmaxx();ymax=getmaxy();cmax=getmaxcolor();
  setviewport(0,0,xmax,ymax,1);
  fprintf(stdprn,"%c%c%c",27,'@');
  /* 打印机恢复 */
  fprintf(stdprn,"%c",17); /* 打开打印机 */
  fprintf(stdprn,"%c%c%c",27,115,0);
  /* 设置消音打印 */
  /* clour-- ①黑色,②品红,③青,④黄,⑤蓝,⑥红,⑦绿 */
  bk=getbkcolor();XM=xmax+1;
  for(i=0;i<=ymax;i+=24)
  { for(ic=0;ic<=cmax;ic++)
    { /* 循环检测[0--cmax]种颜色 */
      kg=0; for(x=0;x<XM;x++)
        for(j=0;j<=2;j++)
          by[j][x]=0x00; /* 数组初始化 */
      for(x=0;x<xmax;x++)
        /* 屏幕水平方向从 0 到 xmax */
        for(j=0;j<=2;j++) /* 垂直方向 3 个字节 */
          for(y=0;y<=7;y++) /* 每个字节 8 位 */
            { cor=getpixel(x,i+j*8+y); /* 打印颜色识别 */
```

```
if(cor==ic){k=1;kg=1;} /* 彩色识别 */
else k=0;
by[j][x]=putbit(by[j][x],k,y);
}
if(! kg) continue;
for(ik=0;ik<3;ik++)
{
  if(ik<8) fprintf(stdprn,"%c%c%c",26,'E');
  /* 设置加重打印 */
  else fprintf(stdprn,"%c%c%c",26,'F');
  /* 取消加重打印 */
  if(col[ik][ic]==0) break;
  fprintf(stdprn,"%c%c%c",27,114,col[ik][ic]%8);
  /* 设置彩色颜色 */
  fprintf(stdprn,"%c%c%c%c",27,42,mode,128,2);
  /* 设置点图象打印模式 */
  /* ESC * [32/33/38/39/40]n1 n2 m1 m2... */
  for(x=0;x<XM;x++)
    for(j=0;j<=2;j++)
      { biosprint(0,by[j][x],0); /* fprintf(stdprn,"%c",by[j][x]); */
      }
  fprintf(stdprn,"%c",13); /* 设置回车 */
  /* 0--15 colour */
  fprintf(stdprn,"%c%c%c",27,51,24);
  /* 打印机行间距(24/180) */
  fprintf(stdprn,"%c%c%c",27,11,1);
  /* 打印机进给一行 */
  fprintf(stdprn,"%c%c",13,10); /* 换行 */
  }
  fprintf(stdprn,"%c",19); /* 关闭打印机 */
  setviewport(v.left,v.top,v.right,v.bottom,v.clip);
  /* 恢复原视见区 */
}
```

本程序在 386DX 微机上用 Turbo C/Borland C++ 系统调试通过,打印机为 CR3240。若用户采用其它彩色打印机,只需稍微修改程序中相应的彩色打印机控制命令即可。

参考文献

- [1]张健,《彩色屏幕的多灰度硬拷贝》,计算机应用研究,1994,N6,p64
- [2]韩龙男,《Turbo C 高分打印函数》,MC,1993,N1
- [3](美)D.F. 罗杰斯著,《计算机图形学的算法基础》,科学出版社,1987 年。

(本文收稿日期:1995.1.6)

通用嵌入式点阵汉字的实现与应用

山东省莱芜市钢城区莱芜钢铁总厂自动化部 隋景明 孙迎春

摘 要:本文以 16×16 点阵的基础点阵为例,说明嵌入式点阵汉字的实现机理与方法,并给出了汉字点阵的程序提取方法,及利用该汉字点阵实现嵌入式点阵汉字显示应用的两个 C 语言程序示例。

关键词:基础点阵,派生点阵,嵌入式点阵汉字,汉字点阵,点阵汉字。

一、引 言

一个好的商用应用软件的设计,往往都很注重软件的封面设计及友好的人机界面。一些演示程序及港台制作的游戏软件等其中的封面设计和人机界面较多地采用了嵌入式的高点阵的点阵汉字的显示及处理技术,从而给人一种赏心悦目的感受。那么我们一般的程序员能否利用这种技术来给我们自己的软件设计增添光彩呢?本文试就这一技术作一简单探讨,并给出在 PC 机中的 C 语言程序事例。

二、实现机理与方法

用这种方法不仅限于汉字,且还可显示用户自定义的汉字,字符图形等。其中的原理是一样的,本文仅以汉字显示为例,加以说明。

首先,确定要显示的汉字及相应的基础点阵,如要显示的汉字串为“莱芜钢铁总厂”,基础点阵为 16×16 。然后产生或形成该汉字串的点阵数据,并嵌入程序,作为程序数据的一部分,供程序调用,而其他的派生点阵,都可以此为基础,产生该字串的放大显示。

汉字点阵数据的生成可以有几种方法,对于 16 点阵或 24 点阵的汉字,可以利用操作系统提供的标准汉字库,用程序自动提取,而对于其他的点阵就不容易得到了,这时,如果汉字数量较少,可以采用手工的方法产生,当然,如果数量较多,就要自己编写相应的造字程序,来产生相应的点阵。至于造字程序,可参考 UC DOS 3.0 中提供的造字程序 MKHZ. EXE。该程序使用方便,形象直观,很值得借鉴。利用它可产生 16×16 的基础点阵。

解决了点阵的产生,有了相应的汉字点阵,汉字的显示就容易实现了。根据点阵数据的存放结构,编写相应的显示程序。标准的 16×16 点阵显示数据一般按行存放,显示程序也需按行处理,即可

显示 16×16 、 32×32 等各种形式的汉字串了。

三、程序事例及运行环境

本文给出 16 点阵汉字的点阵自动提取程序事例,及相应字串的汉字显示实例。程序运行环境为,硬件 IBM PC/AT 286, 386 等系列及其兼容机,软件支持环境为 MS DOS 3.0 以上操作系统,源程序在 Turbo C 2.0 环境下调试通过。程序附后:

```
/* 汉字点阵数据自动提取程序,示例 1 */
/* 程序名称: CREATDZ.C */
#include <conio.h>
#include <fcntl.h>
#include <io.h>
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <graphics.h>
#define LENGTH (32L)
#include <\\tc\\c\\init_gra.c>
void display_hz(int dx, int dy, char *s);
void Initialize();
char *libfile;
main()
{
    page0();
    closegraph();
}

page0()
{
    int leng, i;
    char p3[] = "莱芜钢铁总厂";
    char ss[3];
    ss[2] = '\\0';
    libfile = "c:\\c\\lib16.";
    Initialize();
```

```
printf("\n/ * %s * /\nint static data[][16]={",p3);
for(i=0;i<strlen(p3)/2;i=i+1)
{ss[0]=p3[i*2];ss[1]=p3[i*2+1];
dxdot(ss);}
printf(" ");
getch();
}
dxdot(char *p)
/* 产生汉字的点阵数据;格式为 dat[i][0]=0xdd;dat[i]
[1]=0xdd;... */
{
display_hz(100,200,p);
printf("\n"); }
/* JJ 为全局变量 */
```

```
void display_hz(int dx,int dy,char *s)
{
int i,j,k,data[16],n;
unsigned char qh,wh;
char s1[10];
unsigned int size;
int temp = 0;
int f_handle;
long offset;
int len,skip;
if((f_handle=open(libfile,O_BINARY|O_RDONLY)
)== -1) {
perror("Error:");
exit(0);
}
len = strlen(s);
temp = (int)s[0];
if(temp > 0) skip = 1;
else skip = 2;
for(i=0; i<len; i+=skip) {

qh = s[i];
wh = s[i+1];
temp = (int)s[i];
if(temp > 0) skip = 1;
else skip = 2;
if(temp < 0) {
skip = 2;
qh -= 0xA1;
wh -= 0xA1;
if (qh >=15) qh -= 8;
if((qh==0)&&(wh==0))goto con;
offset=(qh*94L+wh)*LENGTH;
lseek(f_handle,offset,SEEK_SET);
read(f_handle,data,LENGTH);
for(j=0;j<16;j++){
if(j==8) printf("\n");
printf("0x%X",data[j]);
}
}
else {
skip = 1;
```

```
sprintf(s1,"%c",s[i]);
outtextxy(dx+i*8,dy+4,s1);
}
}
con :
close(f_handle);
}
```

```
/* 一个嵌入式小点阵字库的应用程序,示例2 */
/* 程序名称: demo.c */
/* 莱芜钢铁总厂 */
int static data[][16]=
{0x660,0x664,0x7FFE,0x760,0x190,0x1FF8,
0x9A0,0xFB0, 0x5E0,0x3FFC,0x7C0,0xDB0,0x1998,
0x318E, 0x4184,0x100, 0x630,0x632,0x7FFF,
0x630,0x628,0x1FFC,0x180,0x180,0x7FFF,
0x180,0x380,0x7C0,0xCC2,0x18C3,
0x30FF,0x407E,0x1101,0x1DFF,0x1983,0x3F8B,
0x31CF,0x21AF,0x7DBB,0x5993,0x19BB,0x7FEF,
0x19CB,0x1983,0x1B83,0x1D8B,
0x3907,0x1202,0x1020,0x1930,0x11B0,0x3FFE,
0x31B0,0x2330,0x7E30,0x5BFF,0x1830,0x7C68,
0x1868,0x18CC,0x1ACC,0x1D86,0x3B07,0x1402,
0x818,0x630,0x1264,0x1FFE,0x180C,0x180C,0x180C,
0x1FFC, 0x1008,0x100,0x14C8,0x3664,0x3656,
0x7613,0x27FA,0x3F0, 0x1002,0x1FFF,0x1800,
0x1800,0x1800,0x1800,0x1800,0x1800,
0x1800,0x1800,0x1800,0x3000,0x3000,0x2000,0x4000,
};
#include<graphics.h>
void Initialize(void);
main()
{ int i;
Initialize(); /* 图形初始化 */
done();
getch();
}
done()
{int i;
register int color;
int ik,jk;
ik=4;jk=3;
for(i=0;i<6;i++){
COLOR_dzhzj(100+i*16*ik,320,ik,jk,data[i]);
ik=1;jk=1;
for(i=0;i<6;i++){
COLOR_dzhzj(300+i*16*ik,100,ik,jk,data[i]);
/* 利用点阵数据写一个汉字函数 */
}
}
/* 图形初始化子程序 */
#include<graphics.h>
void Initialize(void)
{
int xasp, yasp;
int GraphDriver, GraphMode,ErrorCode;
GraphDriver = DETECT;
(下转 39 页)
```

用五笔字型为 AutoCAD 输入汉字

空军导弹学院机械基础教研室[713800] 兰文祥

AutoCAD 是功能很强的图形软件,五笔字型汉字输入技术是我国普遍使用的汉字输入方法。使用汉化版 AutoCAD 时,存在的问题是:①汉化是针对某种显示器进行的,显示器不同则不能运行。②汉化版占用较多内存,降低了图形处理速度。③专业人员使用 AutoCAD 处理的汉字是有限的。因此,我们可以利用 AutoCAD 提供的软件开发工具 AutoLisp 语言,汉化版的矢量汉字库,建立起 AutoCAD 中汉字的五笔字型输入方法。程序如下:

```
(defun C:WBX()
  (setq hzodelist '(
    ("PV" "安") ("RPV" "按") ("PVS"
"案") ("MMGD" "凹") ("WTY" "八")
    ("RDC" "拔") ("RCN" "把") ("RRRR"
"白") ("RLFC" "摆") ("GYTG" "班")
    ("TEMC" "般") ("SRCY" "板") ("UF"
"半") ("LW" "办") ("QN" "包")
    ; * * * * * 16 * * * * *
    ("SSTL" "机械制图")
    ; * * * * * 词组 * * * * *
  ))
  (print "      WBX Loaded")
)

(defun wait()
  (setq waittime 100)
  (while (> waittime 0)
    (setq waittime (- waittime 1))
  )
)

(defun delent()
  (setq delentname (entlast))
  (entdel delentname)
)

(defun C:HZ()
  (setq hzpt (getpoint "startpoint: "))
  (print)
  (setq hzpt0 hzpt)
```

```

  (setq sblipl (getvar "BLIPMODE"))
  (setq scmde (getvar "CMDECHO"))
  (setvar "BLIPMODE" 0)
  (setvar "CMDECHO" 0);
  (setq hzh (getreal "Height <1.00>: "))
  (if (= hzh nil) (setq hzh 1.00))
  (print)
  (setq hzw (getreal "Width factor <1.00>: "))
  (if (= hzw nil) (setq hzw 1.00))
  (print)
  (setq hza (getreal "obliquing angle <0>: "))
  (if (= hza nil) (setq hza 0))
  (print " * * * type exit to end HzInput * * *")
  (print)
  (setq outstr "")
  (setq gethz (getstring "HzWbxCode: "))
  (print)
  (setq gethz (strcase gethz))
  (while (/= gethz "EXIT")
    (if (= gethz "ASC")
      (progn
        (setq textstr (getstring T "Text: "))
        (setq outstr (strcat outstr textstr))
        (command "text" hzpt hzh hza textstr)
        (wait)
        (delent)
        (setq strl (strlen textstr))
        (setq hzpt (list (+ (* strl hzh hzw) (car hzpt))
(cadr hzpt)))
        (setq gethz (getstring "HzWbxCode: "))
        (print)
        (setq gethz (strcase gethz))
      )
      (progn
        (setq hzlist (assoc gethz hzodelist))
        (if (/= hzlist nil)
          (progn
            (setq hzz (cadr hzlist))
            (setq outstr (strcat outstr hzz))
            (command "text" hzpt hzh hza hzz)
            (wait)
            (delent)
          )
        )
      )
    )
  )

```

```

(setq gethz (getstring "HzWbxCode: "))
(print)
(setq gethz (strcase gethz))

(setq strl (/ (strlen hzz) 2))
(if (= gethz "DEL")
    (progn
        (setq outstrl (strlen outstr))
        (setq strl (* 2 strl))
        (setq outstr (substr outstr 1 (- outstrl strl)))
        (setq strl 0))

    (setq gethz (getstring "HzWbxCode: "))
    (print)
    (setq gethz (strcase gethz))
    )
)

(setq hzpt (list (+ (* strl hzh hzw) (car hzpt))
    (cadr hzpt)))
)
(progn
    (print " * * * * * Not found * * * * * ")
    (print)
    (setq gethz (getstring "HzWbxCode: "))
    (print)
    (setq gethz (strcase gethz))
    )
)

```

使用注意:

1. ACAD 目录中必须存在 HZTXT.SHX 文件。
2. 用 STYLE 命令使当前字型包含 TXT, HZTXT 两种文本的设置。
3. 用 SET 命令为 AutoCAD 运行设置环境, 以使 Autolisp 有足够的运行空间。
4. 用 (load "文件名") 调入。
5. 键入 WBX, 建立起索引表。
6. 键入 HZ, 执行该命令进行汉字标注, 按提示操作。转输入西文时 键入 ASC, 按提示操作。
7. 如有重码, 需对程序作适当修改。

本程序在长城 386 微机, 用 AutoCAD2.6 版运行通过。

(本文收稿日期: 1994.12.19)

通用嵌入式点阵汉字的实现与应用

(上接 37 页)

```

/* 请求自动侦探功能 */
initgraph( &GraphDriver, &GraphMode, "" );
ErrorCode = graphresult();
/* 返回初始化结果 */
if( ErrorCode != grOk ){
    printf(" Graphics System Error: %s\n",
grapherrormsg( ErrorCode ) );
    exit( 1 );
}

COLOR_dzhzj(int dx,int dy,int ik,int jk,int data[16])
{int j,k,n;
int i=0;int color;

```

```

color=getcolor();
setfillstyle(SOLID_FILL,color);
for(j=0;j<16;j++){
    setfillstyle(SOLID_FILL, j);
    for(k=0;k<16;k++){
        n=data[j]; n=n<<k; n=n>>15;
        if(n) bar(dx+i*8*ik+k*ik,dy+j*jk,dx+i*8*
            ik+k*ik+ik-1,
            dy+j*jk+jk-1);
    }
}
}

```

参考文献

- [1] 林亨利, 陈维兴编译, 清华大学出版, Turbo C++ 应用教程(上, 中, 下), 1992 年 1 月
- [2] 张福炎, 徐福培等编注, 南京大学出版, IBM PC 的原理及应用(图形显示器及程序设计, 1990 年 7 月
- [3] 美国 Kris Jamas 著, 《DOS The Complete Reference》

(本文收稿日期: 1995.1.3)

Gene 病毒的原理及其防治

浙江大学信电系物电 94 研 卢胜文

摘 要: 本文主要探讨文件型病毒 Gene 的基本工作原理, 以及其预防和消除方法。

关键字: 文件型病毒, 密钥, 反跟踪, 加密, 解密。

笔者在毕业设计时发现 Gene 病毒, 这是一种国产的文件型计算机病毒, 用国外的 Scan115.Cpav2.0 等杀毒软件都不能发现和清除它, 用公安部发行的 Kill71 也无济于事。计算机感染 Gene 病毒后有明显特征: 使用 dir 命令列目录时, 速度明显变慢, 并有频繁的读写盘操作。病毒发作时, 每天上午四点后, 每隔一分钟就在显示器的左上角显示红色的“Gene!”(其中文意思为“基因”)。

Gene 病毒只感染 .EXE 文件, 文件一旦被感染将增长 1366~1382 字节, 但在带毒环境下看不出文件长度变化。观察一个被感染的文件尾部, 就会发现其最后两个字节为 0xaaaa, 并且在附近还有一个串“Gene __ 1991 __ in DUT (Dalian China)”字符。

Gene 病毒的激活过程: 当运行一个染毒的 .EXE 文件时, 首先运行位于程序尾部的病毒代码, 将病毒代码驻留在内存高端, 并修改 MCB 链, 使其内存减少 1638 字节, 再修改 0x21 中断, 使其指向病毒代码, 这样该病毒就被激活了。这时运行 dir 命令, 就会感染该目录下的 .EXE 文件, 但对有只读属性或以 0xaaaa 结尾的文件不感染。

该病毒采用了多种反跟踪技术, 对其实施动态跟踪时不注意是很容易上当的。该病毒修改了 INT 3 中断, 并把 sp 指针移向一关键数据区附近。若随便用 Debug 来实现断点动态跟踪, 就会把解密的密钥破坏掉。

Gene 病毒的具体传染过程: 首先判断所列的文件是否是 .EXE 文件, 如是则将文件末尾的两个字节读出, 判断是否是 0xaaaa, 若不是则开始感染该文件, 将 .EXE 文件头的 cs、ip、ss、sp 的值保存在偏移病毒代码首部为 0x3f0

处, 并将 cs、ip、ss、sp 的值改为指向病毒代码, 然后再对保留的原 cs、ip、ss、sp 进行加密, 并将病毒开始部分进行解密(病毒开始部分代码附在 .EXE 文件尾时是明文, 但在内存中时是密文), 最后再将加密程序和密钥进行进一步加密, 并把其加在该 .EXE 文件尾部。

由此可见, 该病毒采用了较强的保密技术, 无论是在内存中还是在程序尾部不是全明文。病毒解密的密钥和解密程序都被加过密, 其密钥也是不固定的, 其值从内存 0000:064c 处取得, 并存于病毒代码中。

通过上面的介绍, 我们不难看出防治和清除该病毒的方法, 要防治 Gene 病毒可以将 .EXE 文件改为只读, 或在文件末尾加 0xaaaa。要清除该病毒, 其关键在于恢复 .EXE 文件头的 cs、ip、ss、sp 值。消毒程序清单如下, 该程序清单只是检索和消毒部分, 只要再加上目录扫描部分, 就能完成检测和消毒功能。笔者有完整程序, 它能检测和清除 Gene 和 Dabi 两种病毒(包括单独感染和交叉感染), 由于采用窗口菜单驱动, 程序太长不便摘录。该程序由 Blordland C++ 3.1 编写而成, 并且在 Novell 486/386 系统上运行通过, 在其它微机上也运行良好。清除 Gene 病毒和 Dabi 病毒清单, 它能清除两种病毒的交叉感染:

```
# include<stdio. h>
# include<io. h>
# include<dos. h>
# include<conio. h>
# include<fcntl. h>
# include<stdlib. h>
# include<mem. h>
# include<bios. h>
# include<string. h>
# define __ESC 0x001b
struct exe
{unsigned int linkname;
```

```
unsigned int filelength;
unsigned int filesize;
unsigned int none1;
unsigned int filetop;
unsigned long none2;
unsigned int ss;
unsigned int sp;
unsigned int none3;
unsigned int ip;
unsigned int cs;
}exe;

static char flag1[] =
{"\xeb\x03\x90\xaa\x8c\x02\xe3\x78\x01"};
static char flag[] =
{"\x47\x46\x65\x46\xe4\x65\x42\x42\x21\x4a\xaa"};
static char dabi[] = {"L4QODGDIIB + GLO"};
unsigned char buf[532], buf1[15], y=1;
char killGene(char * fname, long ofs1, long ofs2);
char killDabi(char * fname, long ofs, char exe);
char scandisk(char * fname, char selectkill);
char testexecfile(char * name);
void scanram(void);
void scanram(void)
{unsigned far * m1;
unsigned m;
char far * m2, far * m3, far * m4, i;
m1=(unsigned far *)MK __FP(0, 0X84);
m=* (m1+1);
m2=(char far *)MK __FP(m, 0x5b5);
m3=(char far *)MK __FP(m, 0x648);
m4=(char far *)MK __FP(m, 0x5a8);
buf[31]=0;
for(i=0; i<31; i++)
buf[i]=* (m2+i);
for(i=0; i<14; i++)
buf1[i]=* (m3+i);
buf1[14]=0;
if(! memcmp(buf, "Gene __ 1991 __ in DUT (Dalian China)", 31))
{sound(1000); delay(100); nosound();
printf("Found a [Gene] virus in memory");
if(getch() == __ESC) exit(0);
}
for(i=0; i<14; i++)
buf1[i]=* (m4+i);
```



```

if(memcmp(buf1,dabi,14))
    return;
sound(1000);delay(100);nosound();
printf("Find a [Dabi] virus in memory");
if(getch() == __ESC)exit(0);
}

char killDabi(char * fname, long ofs, char
exe)
{
    struct exe * ep;
    unsigned long tell;
    struct ftime * time;
    unsigned char c, top[0x18], i, j, temp, fattr;
    unsigned char changeflag=0, kill=0;
    int handle;
    memset(top, '\x0', 18);
    fattr = __chmod(fname, 0);
    temp = 0x0f;
    if(temp != 0)
    {
        __chmod(fname, 1, 0);
        changeflag = 1;
    }
    if((handle = open(fname, O__RDWR|O__
        BINARY)) != -1)
    {
        getftime(handle, time);
        if(exe == 1)
        {
            lseek(handle, 0L, SEEK__SET);
            read(handle, top, 0x18);
            lseek(handle, ofs, SEEK__END);
            read(handle, &top[0x0e], 0x0a);
            for(i=0x0a, j=0x0e; i!=0; i--, j++)
            {
                top[j]^=i;
                tell=lseek(handle, 0L, SEEK__END);
                tell-=0x5b9L;
                ep=(struct exe *)top;
                ep->filesize=tell/0x200L+1;
                ep->filelength=tell%0x200L;
                lseek(handle, 0L, SEEK__SET);
                write(handle, top, 0x18);
            }
        }
        else
        {
            lseek(handle, ofs, SEEK__END);
            read(handle, top, 5);
            for(i=5, j=0; i!=0; i--, j++)
            {
                top[j]^=i;
                lseek(handle, 0L, SEEK__SET);
                write(handle, top, 5);
                tell=lseek(handle, 0, SEEK__END);
                tell-=0x5b9L;
            }
            if(! chsize(handle, tell))
            {
                kill=1;
            }
            if(changeflag)
            {
                __chmod(fname, 1, fattr);
                setftime(handle, time);
                close(handle);
            }
            else
            {
                printf("Cann't write diskette");
                return(kill);
            }
        }
    }
}

char killGene(char * fname, long ofs1, long

```

```

offs2)
{
    unsigned long fsize, tell;
    char c[2], str[14], i, temp;
    char changeflag=0, fattr;
    unsigned char data[8], kill=0;
    struct ftime * time;
    unsigned * dp;
    int handle;
    fattr=__chmod(fname, 0);
    temp=fattr;
    temp&=0x0f;
    if(temp!=0)
    {
        __chmod(fname, 1, 0);
        changeflag=1;
    }
    if((handle=open(fname, O__RDWR|O__
        BINARY)) != -1)
    {
        getftime(handle, time);
        lseek(handle, 0L, SEEK__SET);
        memset(&exefile, '\x0', sizeof(struct
            exe));
        read(handle, &exefile, sizeof(struct
            exe));
        fsize=(unsigned long)(exefile.filesize*
            0x200L)+
            (unsigned long)exefile.filelength;
        fsize-=0x556;
        exefile.filesize=(unsigned int)(fsize/
            0x200);
        exefile.filelength=(unsigned int)(fsize%
            0x200);
        tell=lseek(handle, 0L, SEEK__END);
        lseek(handle, ofs1, SEEK__END);
        read(handle, c, 2);
        lseek(handle, ofs2, SEEK__END);
        read(handle, data, 8);
        read(handle, str, 14);
        c[0]=c[0]^c[1];
        for(i=0; i<8; i++, i++)
            data[i]^=c[0];
        dp=(unsigned *)data;
        exefile.cs=*dp;
        exefile.ip=*(dp+1);
        exefile.ss=*(dp+2);
        exefile.sp=*(dp+3);
        for(i=0; i<14; i++, i++)
            str[i]^=c[0];
        lseek(handle, 0L, SEEK__SET);
        write(handle, &exefile, sizeof(struct
            exe));
        if(! chsize(handle, tell-0x556))
            kill=1;
        if(changeflag)
        {
            __chmod(fname, 1, fattr);
        }
        else
        {
            printf("Write diskette wrong");
            setftime(handle, time);
            close(handle);
            return(kill);
        }
    }
}

char testexecfile(char * name)
{
    int i=0;

```

```

while(* (name+i) != '\x0')i++;
if(! strcmp(name+i-3, "EXE"))
    return 1;
if (strcmp (name + i - 3, "COM")
    &&strcmp (name + i - 3, "
    OV") )
    return 0;
else
    return 2;
}

char scandisk(char * fname, char selectkill)
{
    int i, j;
    char kill1=0, kill2=0, fbuf[448], ft;
    char dabiflag=0, geneflag=1;
    FILE * fp;
    if(! (ft=testexecfile(fname)))
        return 0;
    if((fp=fopen(fname, "rb")) != NULL)
    {
        j=0;
        fseek(fp, -420L, SEEK__END);
        fread(buf, sizeof(unsigned char), 420,
            fp);
        while(memcmp(&buf[j], dabi, 14)&&(j
            <418))j++;
        if(j<418)
            dabiflag=1;
        i=0;
        fseek(fp, -430L, SEEK__END);
        memset(buf, '\x0', 432);
        fread(buf, sizeof(unsigned char), 430,
            fp);
        while(memcmp(&buf[i], flag, 14)&&(i
            <419))i++;
        if(i==419)
        {
            fseek(fp, -0x6f6L, SEEK__END);
            memset(fbuf, '\x0', 428);
            fread(fbuf, sizeof(char), 427, fp);
            fclose(fp); i=0;
            while(memcmp(&fbuf[i], flag1, 11)&&
                (i<420))
                i++;
            if(i==420)
            {
                if(! dabiflag)return 0;
                geneflag=0;
            }
        }
        fclose(fp);
        if(selectkill)
        {
            if(geneflag)
            {
                kill1=killGene(fname, i-400-0xe5, i-
                    400-0x276);
                if(dabiflag)
                {
                    kill2=killDabi(fname, -0x98+j-400,
                        ft);
                }
            }
            if(geneflag)printf("Killed Gene in %s",
                fname);
            if(dabiflag)printf("Killed Dabi in %s",
                fname);
            return 0;
        }
        return -1;
    }
}

```

(本文收稿日期:1994.12.4)

Greeting 病毒分析及清除

华中理工大学自控系(430074) 罗 明 朱德森

摘 要 文中描述了一种新发现的文件外壳型病毒,并结合此类病毒共有的特点对其进行了详细的分析,最后给出了检测及清除的方法。

关键词 外壳型文件病毒 病毒检测 系统免疫 文件恢复

前段时间,本系实验室发现一种新的病毒。用 DOS6.0 所带的 MSAV、VIRUSCAN V104 版都不能查出,有关资料也未见报导。因其在表现时显示 "Greeting from cqZGB" 等字样,我们暂且称其为 Greeting 病毒。

通过阅读病毒代码,我们发现此病毒属于最普通的一类外壳型文件病毒,很具代表性。根据病毒数据库 VSUM (93/5) 的资料,在当时发现的 2015 种 DOS 病毒中,文件型病毒占 95.53%,达 1925 种;而其中绝大部分正是这种常见的类型。考虑到它的典型性,我们对此病毒作了详细的剖析。

一、Greeting 病毒特点及分析

此类病毒大多具有相似的结构和特点,所用的手法也大致相同。下面就这些病毒共有的特征对 Greeting 进行分析。

1. 攻击对象

凡未染此病毒的长度大于 6,144 (1800H) 字节的可执行文件皆是它感染的目标。病毒把自身复制到文件末尾,并修改文件头 (EXE 文件) 或文件开头 12 字节 (COM 文件),以使程序加载执行时先执行病毒程序。由于在感染时,病毒先用 INT 21H 的 43H 功能将文件属性改为普通文件,传染完后再行恢复,故无论系统或只读文件都不能幸免,只要磁盘空间够就可能被传染。

2. 文件感染标志

用 MSAV 可以看到,被感染的可执行文件增加了 2128 字节 (实际上是 $850H + X (0 \leq X < 16) \text{ BYTES}$)。文件创立时间的秒钟计数全被置为 30 秒。病毒正是以 DIR 命令以及 PCTOOL 等工具看不到的秒钟数作为文件是否已被感染的标志,防止文件重复感染的。

3. 驻留标志

此病毒驻留内存并截取 INT 21H 中断。它修改 30H 号功能调用作为自身已驻留内存的标志以避免重复驻留。具体说就是:如果以 $BX = 9999H$ 的入口参数调用取 DOS 版本号功能,AX 将返回 9999H 标志而不是 DOS 版本号;否则一切正常。据说另有一种病毒采用了几乎一样的手法,是巧合还是另有渊源不得而知,但这种设标志的方式是典型的和常用的。不驻留内存的病毒则没有此标志。

4. 传播机制

病毒复制自己总是在正常的磁盘操作前后进行,以免为人所觉。Greeting 病毒就采用了这类病毒常用的几种方式:

第一,截取 11H、12H 功能调用,在执行 DIR 命令、进行目录搜索时传染文件。病毒记下搜索到的第一个符合传染条件的可执行文件的路径及名称,在 12H 功能调用失败即目录搜索完成时将此文件加上一层病毒外壳。仔细观察可以发现,在带毒系统的硬盘子目录上执行 DIR 命令时,最后往往有一个明显的停顿,这正是病毒在进行传染。

其次,病毒修改了 3CH、3EH 功能。系统执行句柄文件打开操作时,如果文件符合感染条件则记下文件的句柄号;以后系统调用 3EH 功能关闭此文件时把它染上毒。在同时打开了几个文件的情况下,病毒只对符合条件的第一个文件进行感染。

另外,它修改了 4BH 功能,在系统加载文件时感染所加载的可执行文件。

所有这些传播方法一次只能传染一个文件。这是因为给文件加外壳的传染方式磁盘文件操作量大,时间长,一次传染多个文件容易引起用户警觉。相比之下,DIR-II 一类的病毒采用修改

FAT 表磁盘链簇的办法,磁盘操作少,可以一次传染整个目录。

以上的几种传播方式虽说效率不高,但由于机会频繁,此病毒的传染及扩散速度还是较快的。

5. 伪装手法

值得一提的是,此病毒引入了一些精心设计的伪装,竭力隐藏自己活动的种种迹象。

第一,在截取 11H、12H 中断进行传染的同时,病毒检查 DTA 中的一个含有目录项信息的标准结构,将已感染的文件的长度减少 850H 字节。这样,在带毒的系统中用 DIR 命令看到的被感染文件的长度并未增加。病毒掩盖了文件变动的迹象,企图以此逃过用户的眼睛。

第二,截取 35H、25H 号功能调用,使 35H 功能被用来取 INT 21H 的中断向量时,返回的是指向以往的正常的 INT 21H 的中断向量;而使用 25H 功能截取 INT 21H 中断时,病毒记下新的中断处理的入口地址,并不真正修改系统中断向量,而是在自身执行完后转去执行这个新的中断处理。通过这一手法使自己尽可能控制着 INT 21H 并躲过某些侦毒软件的检查。

第三,病毒在进行传染时,先设自己的严重错误处理中断,然后再进行磁盘文件操作,结束时又将 INT 24H 予以恢复,以此避免正常的错误处理程序产生的提示暴露病毒传染文件的行为。

具体说来,Greeting 的 INT 24H 中断只有两行代码:

```
XOR AL,AL
IRET
```

当出现严重错误时,系统将根据 INT24H 返回的 AL 值选择处理错误的方法;而 AL=00 表示忽略此错误,所

以系统将不产生提示继续执行。但由于其处理过于简单,没考虑 DOS 对许多错误是不能忽略的这一条,所以往往仍有执行不正常的迹象泄漏出来。

最常见的情况是:带毒执行“DIR A:”而 A 盘写保护时,病毒试图感染 A 盘文件;如果不修改 INT 24H, DOS 将显示:

```
Write protect error writing drive A
Abort, Retry, Fail?
```

病毒的行为将暴露无疑。而在病毒伪装机制的作用下,则出现“Fail on INT 24”或“Extended Error 83”等出错信息。由于提示含糊不清,用户往往怀疑软驱有毛病而没想到这正是病毒在作祟。

6. 激活与表现

从设计的角度来说,病毒的表现与破坏部分受的约束较少,其作者可以自由发挥,奇思百出,差别较大。

Greeting 病毒平时只是传染文件,但在 4 月 17 日及 8 月 17 日将出来表现一番。这两天,如果执行 DIR 命令,病毒将删除它找到的第一个符合感染条件的可执行文件,用户往往发现执行一次 DIR 就少了一个文件;同时,执行任一可执行文件,此文件将被病毒删除,结果出现“Bad command or file name”的出错提示。但 Greeting 病毒恶作剧的味道多于存心的恶意,在总共删除了三个文件后就停止破坏,转而在执行 DIR 或 DOS 外部命令时执行其表现部分,在屏幕正中用红底蓝字(带闪烁)显示如下字样:

```
Aha! Greeting from cqZGB,
can you find me? Bye-bye!
几秒钟后跳转到 ROM BIOS 的 F000:
FFF0,重新启动。
```

二、病毒的检测与清除

1. 文件中病毒的检查

用 DEBUG 检查感染后的 COM 文件,将发现开头有 12 个字节被修改成如下一段代码:

```
XXXX:0100 8CCA MOV DX,CS
XXXX:0102 81C2B08 ADD DX,088B
XXXX:0106 52 PUSH DX
XXXX:0107 BA0001 MOV DX,0100
XXXX:010A 52 PUSH DX
XXXX:010B CB RETF
```

而被感染的 EXE 文件开头则是 JMP 017E;稍一跟踪即可发现病毒判断内存驻留标志的一段代码:

```
XXXX:017E B430 MOV AH,30
```

```
XXXX:0180 BB9999 MOV BX,9999X
XXXX:0183 CD21 INT 21
XXXX:0185 3D9999 CMP AX,9999
XXXX:0188 7506 JNZ 0190
.....
```

另外,病毒表现时的显示信息加密后存于文件末尾。由于所用的加密方式仅仅是简单地与固定密钥 55H 异或,所以仍可作为病毒的特征代码供查毒杀毒程序使用,手工用 DEBUG 查毒时可用 S 命令搜索此段密文:

```
75 14 3D 34 74 75 12 27 30 30 21
3C 3B 32 75 .....
```

如果有以上内容即可断定存在 Greeting 病毒。

用 MSAV 等工具也容易发现此病毒对文件感染的迹象。而 VIRUSCAN 等可以临时添加新病毒签名(代码)以检查有无该种新病毒的反毒工具使用起来则更为直接,具体用法参见有关说明。

2. 内存中病毒的检查及系统免疫

利用病毒的驻留标志可以很容易地检查内存中的病毒并使系统对此病毒免疫。依笔者的心得,对付此类病毒编制疫苗是一个快捷有效的办法,在遏止其传播和破坏、保证工作正常进行以后,再可以抽出充分的时间将其消除(参见附 1)。

3. 病毒的清除

一般来说,病毒为了保证程序的正常执行将保存被感染文件的完整信息,据此我们找到了病毒恢复执行程序的代码,并编出了可全盘扫描的专门对付 Greeting 病毒的杀毒程序。由篇幅所限,在此不给出源程序,感兴趣的可与作者联系。

外壳型病毒感染文件的方式千篇一律。目前虽已见到不修改中断向量而修改内存中中断处理程序前几行代码的病毒,但还尚未见到以不修改文件开头代码的方式传染文件的外壳型病毒。所以对此类中不同的病毒而言,恢复被感染文件、脱去病毒外壳的杀毒工作基本类似,只是具体的文件恢复过程有差别而已。

三、总结

从以上的分析可以看出, Greeting 病毒十分完整地具有外壳型病毒所具有的各种特点和各个组成部分,所用的手法也是“经典”的。对它进行解剖有利于对此类病毒的认识。

单从病毒编程上的特点来看,它没有对自身加密,代码较为好读;也没有较强的反跟踪措施:虽然设置了些障碍,如封锁键盘、封锁中断、修改 00~03H 中断对付动态跟踪、代码段址移动以抗静态分析、以较隐蔽的段溢出等手段让人迷惑等,但都属于常见的小技巧,易识破,也好对付。从这点以及病毒作者的签名(cqZGB)等迹象来看,我们推测此病毒出自国内。

由于专门的反毒软件的广泛使用,病毒的生存空间已压缩得比前两年小得多,国外流行的病毒绝大多数会有商业化的反毒软件检测出来(例如, VSUM (93/5)同时期的 VIRUSCAN V102 版可查病毒 1977 种,占已知病毒的 93.2%)。所以笔者以为,今后实际的反毒工作着重要对付的可能正是这类结构相似甚至雷同,但版本繁多、层出不穷的“土生土长”的病毒。因为这是国外商业化反毒软件顾及不到的空白点;而且病毒在国外发展的过程也预示了这一趋势。但只要我们对此类病毒有着充分的认识,并密切注意病毒技术发展的动向,完成这一任务应该是不困难的。

附 1. 系统内存检测及免疫程序源文件:

(MASM 4.0 汇编, SUPER 386 上运行通过)

```
SEG-A SEGMENT
ASSUME CS: SEG-A, DS:
SEG-A, ES: SEG-A
ORG 100h

START:
PUSH CS
POP DS
PUSH CS
POP ES
MOV AX, 3000H
MOV BX, 9999H
INT 21H
CMP AX, 9999H
JNZ NEXT
CMP BX, 6666H
JNZ FIND-IT-now
MOV DX, OFFSET Aready-in
MOV AH, 09H
INT 21H
JMP HOME
```

FIND-IT-now:

(下转 47 页)

PC 机的键盘及其编程

中国科技大学 王 革

摘 要: 本文从键盘监控、键盘缓冲区、重编键盘中断三个方面系统地介绍了有关 PC 机键盘的知识,以及设计键盘处理程序时应注意的问题。还给出了一个可以随时弹出、进行键盘宏定义的内存驻留(TSR)程序作为示例。

关键词: 键盘监控、键盘缓冲区、键盘中断(ISR)、内存驻留程序(TSR)

一、键盘监控

微机的键盘有一个微处理器,它读入每个键入字符,并将扫描码存于端口 60H。扫描码占一个字节,低 7 位是键的数字编码,与键盘上的键一一对应。最高位为 1 时,表示键被按下,叫通码;最高位为 0 时,表示键被释放,叫断码。同时,微处理器还调用键盘中断 INT 9。INT 9 中断服务例程分析扫描码,当扫描码来自组合键(Ctrl, Alt, Shift)或双态键(Insert, CapsLock, NumLock, ScrollLock)时,这些键的状态变化就被记录在主存 0040:0017 (Hex)处的一字节中。这一字节各位为 1 时的意义如下:

- | | |
|-------------------|----------------------|
| 7: Insert 键处于 on | 6: CapsLocks 键处于 on |
| 5: NumLock 键处于 on | 4: ScrollLock 键处于 on |
| 3: Alt 键被按下 | 2: Ctrl 键被按下 |
| 1: 左 Shift 键被按下 | 0: 右 Shift 键被按下 |

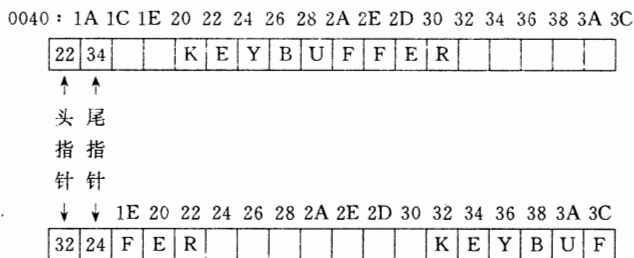
其中,0—3 四位仅当对应键被按下且未释放时才为 1,一旦对应键被释放,又立即变回 0。用程序改变这些位,将等效于敲入对应键。例如,程序可以置 NumLock 和 CapsLock 状态位为 1,以确保按要求输入。

当扫描码来自组合键及双态键以外的键时,键盘中断服务程序将扫描码变换成字符码存入键盘缓冲区。当然,首先要检查组合键或双态键的状态,以获得正确的字符码(如是'a'还是'A'?)。字符码占两个字节。Esc, Tab, Backspace, Return, 字母键, 数字键及符号键所产生的字符码,第一字节是 ASCII 码,第二字节是该键的扫描码;而光标键,功能键(F1—12)及 Insert, Delete, End, Home, PageUp, PageDn 等键产生的字符码,第一字节是 0,第二字节是扫描码;“ALT + 字母”的字符码第

一字节是 0,第二字节字母键的扫描码;“Ctrl + 字母”的字符码第一字节是 0,第二字节是字母的 ASCII 码;组合键及双态键除 Insert 外均不产生字符码。

二、键盘缓冲区

键盘缓冲区位于主存 0040:001C — 0040:003C,共 32 个字节。它是一个循环队列,队列的头指针存放于 0040:001A 处的一个字(2 字节)中,尾指针存放于 0040:001C 处的一个字中。头指针指示缓冲区中第一个键入字符的位置,尾指针指示最后一个键入字符后面的位置,它们的值均在 1CH — 3CH 间变化。键入新字符时,将新字符码放在尾指针指示的地方,尾指针随之改变。BIOS Int 16H 的 00H 号功能调用是“读下一个键盘字符”,实际上就是将缓冲区队头字符返回给 AX,并改变队列头指针。头、尾指针的变化都是每次加 2,超过 3CH 则变为 1CH。而头尾指针相等,表示缓冲区为空。由于尾指针指示的位置是虚设的,总是放着 Return 键的 ASCII 码和扫描码,所以键盘缓冲区最多容纳 15 个字符。缓冲区满后,再输入的字符会丢失,且扬声器发出嘟嘟声。假设键盘缓冲区中含有刚刚键入的字符串“KEYBUFFER”,下图是可能出现的两种情况:



用程序向缓冲区中插入字符可以实现一些有趣的功能。例如,在程序结束前向缓冲区里插入'D', 'I', 'R', 'Return', 那么当控制回到 DOS 后,会自动执行

DIR 命令。

三、重编键盘中断

一般,一个键被按下或释放时均产生一次 INT 9 中断。ROM BIOS 中的 INT 9 中断服务程序 (ISR) 执行以下两个功能:

1、从端口 60H 读取扫描码,并将应答信号送到键盘微处理器;2、根据扫描码产生字符码存入键盘缓冲区或设置组合键/双态键状态标志位。

一般来说,用户重编的 INT 9 ISR 并不完全取代原 INT 9 ISR,而只是截获自己关心的键,其余键则调用原 ISR 处理。新的 ISR 可通过从端口 60H 读取扫描码来判断所击的键,从而决定程序走向。在没有调用原 ISR 的分支中,必须完成将应答信号送到键盘微处理器的工作,否则会导致键盘死锁。把端口 61H 第 7 位置 1,再立即置 0,就将应答信号送到键盘微处理器了。另外,还必须向端口 20H 送数 20H,清除硬件中断。这些可用以下几条 Turbo C 语句完成:

```
char port61;
port61 = inportb( 0x61 );
port61 |= 0x80;
outportb( 0x61, port61 );
port61 &= 0x7f;
outportb( 0x61, port61 );
outportb( 0x20, 0x20 );
```

用户重编的 INT 9 ISR 也可以一开始就保存键盘缓冲区尾指针的值,再调用原 INT 9 ISR (这使得键盘缓冲区尾指针的值发生了变化),然后从键盘缓冲区里取出最后一个字符码,据此判断所击的键。如果是需要截获的键,且希望该键除激活相应功能外无其它副作用,则执行相应功能后,应恢复键盘缓冲区尾指针为原来的值。

四、程序示范

上机时常会碰到需要重复相同击键序列的情况。如果能将这一击键序列用击一个键代替,无疑会方便不少。用一个键来等效于多个键就叫键盘宏定义。例子程序 keymacro.c 是一个内存驻留 (TSR) 程序,功能是让用户将小键盘上的 1, 2, 3 三个键随时定义为所需的三个击键序列,并且能随时改变这三个键的定义。这里的击键序列,可由一般字符键,组合键及特殊键组成,这就使被定义的键不仅能代替一串字符的输入,还能代替选择菜单项,长距

离移动光标等需多次按键才能完成的控制功能。

Ctrl 和 Alt 一齐按下可激活该 TSR。之后按一下想要定义的那个键 (小键盘上 1, 2, 3 三个键中的一个),即可输入要定义的按键序列,再按一下被定义的那个键就结束定义。这里的按键序列不得多于 12 个键。

程序的设计思想是:自编的 INT 9 ISR 把用户定义某键时输入的按键序列所产生的字符码序列保存,此后当定义过的键被按下时,就往键盘缓冲区里送入相应字符码序列,从而达到等效的目的。

程序用 Turbo C 2.0 编写,在 Super 286, AST 386 上运行成功。

以下是程序 KEYMACRO.C:

```
#include <dos.h>
#include <conio.h>
#define KeyBoardInt 9
#define ESC 1
#define Key __1 2
#define Key __2 3
#define Key __3 4
#define Attr 0x7900

int ( far * screen )[ 80 ];
void interrupt ( * OldKey )( void );
enum Status { Normal, SelectKeyToDefine, Define };
enum Status status = Normal;
int DefinedStr[ 3 ][ 15 ] = { { 0 }, { 0 }, { 0 } };
int OldScreen[ 9 ][ 30 ];
char Input [ ] = " Please input and press ";
char HowToExit [ ] = " selected key to exit. ";

unsigned char EndDefineScanCode;
unsigned char CtrlAltStatus;
int i, j;
int KeyNo ;
unsigned char PreIntScanCode = 0xff;
unsigned char Port61;
unsigned char KeyBufTailPtr, OldKeyBufTailPtr;
unsigned int KeyCodeFromBuf, DefinedStrPtr;
unsigned char KeyCodeFromPort;
unsigned char * SelectScreen [ ] =
{
    "===== ",
    " ",
    " 1: ",
    " 2: ",
    " 3: ",
    " Please chose 1,2 or 3 ",
    " on the keypad, or ESC ",
    " ",
    "===== "
};
```

```

void SaveScreen()
{
    for( i = 0; i <= 8; i++ )
        for( j = 0; j < 30; j++ )
            OldScreen[ i ][ j ] = screen[ 10 + i ][ 20 + j ];
}

void ClearScreen()
{
    for( i = 1; i <= 7; i++ )
        for( j = 1; j < 29; j++ )
            screen[ 10 + i ][ 20 + j ] = ' ' + 0X7900;
}

void Select()
{
    for( i = 0; i <= 8; i++ )
        for( j = 0; j < 30; j++ )
            screen[ 10 + i ][ 20 + j ] = SelectScreen[ i ][ j ]
                + Attr;
    status = SelectKeyToDefine;
}

void ClearKeyBoardInt()
{
    static char Port61;
    Port61 = inportb( 0x61 );
    Port61 |= 0x80;
    outportb( 0x61, Port61 );
    Port61 &= 0x7f;
    outportb( 0x61, Port61 );
    outportb( 0x20, 0x20 );
}

void RestoreScreen()
{
    for( i = 0; i <= 8; i++ )
        for( j = 0; j < 30; j++ )
            screen[ 10 + i ][ 20 + j ] = OldScreen[ i ][ j ];
    status = Normal;
}

void interrupt NewKey( void )
{
    KeyCodeFromPort = inportb( 0x60 );
    if( ( KeyCodeFromPort & 0x80 ) ) {
        PreIntScanCode = KeyCodeFromPort;
        OldKey();
        return;
    }
    switch( status ) {
        case Normal:
            KeyCodeFromPort = inportb( 0x60 );

```

```

            KeyBufTailPtr = peekb( 0x40, 0x1c );
            OldKey();
            if( ( KeyCodeFromPort == 0x4f || KeyCodeFromPort == 0x50
                || KeyCodeFromPort == 0x51 ) && PreIntScanCode != 0xe0 ) {
                pokeb( 0x40, 0x1c, KeyBufTailPtr );
                DefinedStrPtr = 0;
                while( DefinedStr[ KeyCodeFromPort - 0x4f ][ DefinedStrPtr ]
                    != 0 ) {
                    poke( 0x40, KeyBufTailPtr,
                        DefinedStr[ KeyCodeFromPort - 0x4f ][ DefinedStrPtr ] );
                    DefinedStrPtr++;
                }
                if( KeyBufTailPtr == 0x3c )
                    KeyBufTailPtr = 0x1e;
                else KeyBufTailPtr += 2;
            }
            pokeb( 0x40, 0x1c, KeyBufTailPtr );
            return;
        }
        CtrlAltStatus = peekb( 0x40, 0x17 );
        if( ( CtrlAltStatus & 0x0c ) != 0x0c )
            return;
        SaveScreen();
        Select();
        break;
    case SelectKeyToDefine:
        KeyCodeFromPort = inportb( 0x60 );
        ClearKeyBoardInt();
        switch( KeyCodeFromPort ) {
            case 0x4f:
            case 0x50:
            case 0x51:
                KeyNo = KeyCodeFromPort - 0x4f;
                DefinedStrPtr = 0;
                status = Define;
                EndDefineScanCode = KeyCodeFromPort;
                for( j = 1; j < strlen( Input ); j++ )
                    screen[ 15 ][ 21 + j ] = Input[ j ] + Attr;
                for( j = 1; j < strlen( HowToExit ); j++ )
                    screen[ 16 ][ 21 + j ] = HowToExit[ j ] +
Attr;
                break;
            case ESC:
                RestoreScreen();
                break;
        }
        break;
}

```



```

case Define;
    OldKeyBufTailPtr = peekb( 0x40, 0x1c );
    KeyCodeFromPort = inportb( 0x60 );
    OldKey( );
    KeyBufTailPtr = peekb( 0x40, 0x1c );
    if( KeyBufTailPtr == OldKeyBufTailPtr )
        return;
    KeyCodeFromBuf = peek( 0x40, OldKeyBufTailPtr );
};

if( KeyCodeFromPort != EndDefineScanCode && DefinedStrPtr != 13 ){
    DefinedStr[ KeyNo ][ DefinedStrPtr ] = KeyCodeFromBuf;
    DefinedStrPtr++;
    screen[ 12 + KeyNo ][ 32 + DefinedStrPtr ] =
        KeyCodeFromBuf % 256 + Attr;
}
else if( KeyCodeFromPort == EndDefineScanCode )
{
    DefinedStr[ KeyNo ][ DefinedStrPtr ] = 0;
    RestoreScreen();
}

pokeb( 0x40, 0x1c, OldKeyBufTailPtr );
break;
}
}

main()
{
    unsigned VideoBaseAdr;
    union REGS regs;
    int86( 0x11, &regs, &regs );
    regs.x.ax &= 0x30;
    if( ( regs.x.ax >> 4 ) == 0x03 )
        VideoBaseAdr = 0xb000;
    else
        VideoBaseAdr = 0xb800;
    OldKey = getvect( KeyBoardInt );
    screen = MK __FP( VideoBaseAdr, 0 );
    disable();
    setvect( KeyBoardInt, NewKey );
    enable();
    keep( 0, 1000 );
}}

```

参考书目:[1]《IBM PC 编程指南》, 电子工业部出版

(本文收稿日期:1994.11.10)

Greeting 病毒分析及清除

(上接43页)

```

MOV DX,OFFSET FIND-IT
MOV AH,09H
INT 21H
JMP HOME
NEXT:
MOV AX,3521H
INT 21h
MOV SI,OFFSET OLDINT21H
MOV [SI],BX
MOV [SI+2],ES
MOV AX,2521H
MOV DX,OFFSET INT-21H
INT 21H
MOV DX,OFFSET MESSAGE
MOV AH,09H
INT 21H
MOV DX,((OFFSET PGMEND
+15)/16)+10H
MOV AX,3100H
INT 21H
HOME:
MOV AX,4C00H

INT 21H
;-----
INT-21H PROC FAR
CMP AH,30h
JNZ RET-21H
CMP BX,9999H
JNZ RET-21H
MOV AX,BX
MOV BX,6666H
RET 2
RET-21H:
PUSHF
CALL CS:OLDINT21H
STI
RET 2
INT-21H ENDP
;-----
OLDINT21H DD ?

FIND-IT DB 07H, 07H, 0Dh,
0Ah, 'Find [Greeting]
virus in memory ! ';
DB 0Dh, 0Ah, 'Please
restart your system...
',0Dh,0Ah,'$'
Already-in DB 'The immuner is al-
ready in system ! ',
07h,'$'
MESSAGE DB 'OK! The immuner
is in system now ! ',
'$'

PGMEND EQU $-start
;-----
SEG-A ENDS
END START

```

参考文献

- [1]《局部网操作系统 DOS 高级技术分析》, 国防工业出版社
- [2]《计算机病毒分析与防治》, 清华大学出版社

(本文收稿日期:1994.11.11)

利用定时器中断预防键盘锁死

安徽财贸学院[233041] 车光宏

摘要:本文分析了死机的原因,给出了预防因键盘中断被关闭而导致的死机的方法。

关键词:预防死机 中断

在计算机上工作,最令人烦恼的事莫过于计算机锁死了。正在输入一篇文稿,突然死机了。无可奈何,只得去按“RESET”,半天辛辛苦苦的劳作付之东流。

然而,不幸的是计算机工作时突然死机的现象经常发生。所以专家们谆谆告诫:要及时存盘。

所谓“死机”,实际上只是键盘被锁死了,而机器并没有“死”。比如,本人有过这样的经历:正在用WPS输入一个源程序,突然键盘失灵,无法用键盘发存盘命令。准备去按“RESET”时,意外地发现鼠标器还灵。于是用鼠标存了盘,挽救了输入的程序。这一事实表明:机器没有“死”,只是键盘被锁死了。

引起键盘锁死的原因有很多,比如电磁干扰、病毒破坏、软件中隐含的错误等都可导致键盘锁死,但归纳起来不外如下两种情况:

1. 键盘中断被关闭

这种情况发生时,键盘被彻底锁死了。此时打任何键、打任意多次,系统均无反应,甚至打Ctrl+Alt+Del键也不能使机器重新启动。

2. 程序不响应键盘中断

出现这种情况时,击键若干次(次数大于键盘缓冲区长度)后会听到笛声,若打Ctrl+Alt+Del键能使机器重新启动。

如果我们能保证键盘中断不被关闭,则可避免因第一种情况而导致的死机。这一任务恰好可利用定时器中断来完成。

定时器中断每秒发生18.2次,DOS为此中断提供的中断服务程序只有一条中断返回指令(IRET),我们用打开键盘中断的操作程序来作为定时器中断的服务程序,即能

保证键盘中断不被关闭。

键盘中断的允许与禁止由端口21H的D1位控制:D1=0,允许;D1=1,禁止。因此,下面程序可打开键盘中断:

```
IN    AL,21H
AND   AL,0FDH
OUT   21H,AL
```

此程序将端口21H的D1位清0(即打开键盘中断),其它位保持不变。以此段程序作为INT 1CH的中断服务程序即可将因某种原因而关闭掉的键盘中断重新打开。

下面程序将打开键盘中断的程序段作为INT 1CH的中断服务程序驻留内存(由于程序很短,故将整个程序一起驻留内存):

```
XXXX:0100  PUSH    CS
XXXX:0101  POP     DS
XXXX:0102  MOV     DX,0112
XXXX:0105  MOV     AX,251C
XXXX:0108  INT     21
XXXX:010A  MOV     AX,3100
XXXX:010D  MOV     DX,0012
XXXX:0110  INT     21
XXXX:0112  PUSH    AX
XXXX:0113  STI
XXXX:0114  IN      AL,21
XXXX:0116  AND     AL,FD
XXXX:0118  OUT     21,AL
XXXX:011A  CLI
XXXX:011B  POP     AX
XXXX:011C  IRET
```

用DEBUG把此程序输入,并取名(比如UNLCKB.COM)存盘。执行一下该程序后进行文本编辑等工作,可大大减少死机的机会。(本文收稿日期:1994.12.21)

好消息

本刊代销 95 年新版 《中国电子行业单位名片集》

10000 多个单位 1000 多页 定价 160 元

由中国电子报社编辑,电子工业出版社出版的《中国电子行业单位名片集》(简称《名片集》)。1995 年版,是在全国电话实现程控化和长途区号三月份重新调整后而修订出版

的。

新版《名片集》刊登的单位名片内容有下列 11 项:单位名称、主要负责人的姓名和职务、详细地址、邮政编码、长途电话区号、电话、电报、传真、电传和 70 个字的主要产品或主要经营项目介绍。

该书共分四大部分:(1)九五百强企业篇;(2)万家名片;(3)新闻出版篇;(4)附录,全国简明邮政编码及最新长途电话区号。

95 年版《名片集》收录了 10000 多个单位,约 1000 多页,每册定价 160 元(免邮寄挂号费)。

欲购该《名片集》者,请速通过邮局汇款到本刊,收购人张纬铮。