文章编号：2096-1472(2016)-03-55-03

嵌入式远程控制系统的设计与实现

王 平，叶福兰，陈章斌
（福州外语外贸学院，福建 福州 350202）

摘 要：随着科学技术的不断发展，嵌入式远程控制系统被广泛应用于各个领域。要远程就必须有通信，要控制就必须有执行，成系统就要内部通信，解决的方案有多种且各有特点。本文以智能家居控制系统为例，从硬件上详细地阐述实现通信模块、执行模块其功能以及内部通信总线的几种方案。最后重点针对使用数据库时如何用C语言编写DBF文件的难题进行软件技术上的解决描述。

关键词：嵌入式；智能家居；C语言编写DBF
中图分类号：TP273 **文献标识码**：A

Design and Implementation of Embedded Remote Control System

WANG Ping,YE Fulan,CHEN Zhangbing
(*Fuzhou College of Foreign Studies and Trade*,*Fuzhou* 350202,*China*)

Abstract: With the continuous development of science,the embedded remote control system is widely used in various fields.Must have the communication,must have the control to carry on,the system must have the internal communication,the solution has many and each has the characteristic.In this paper,a smart home control system for an example, from the hardware details of the communication module,the implementation of its functions and internal communication bus.Finally,to use the database how to use C language problem of DBF file description of software technology to solve.

Keywords:embedded;smart home;C language DBF

1 引言(Introduction)

信息化的21世纪，各种电信和互联网新技术推动了人类文明的巨大进步。在此基础开发出远程控制嵌入式系统的各种应用极大地提高了工作效率和生活便利。智能家居就是嵌入式远程控制的一个具体应用[1]。

多年来，笔者在智能家居方向做了很多开发应用。本文就如何在嵌入式系统上利用现成的电话网和互联网来实现远程控制进行了多种成功方案的硬件描述，这些方案在使用上各有特点、实现的难易程度不同。完成远程通信后，在具体内部控制执行上，由于采用总线结构、自学习式红外无线的方案，可以很好地满足各种现场环境和扩展的需要。

而这些众多的智能终端又可以组成更大的系统，此时就需要用到数据库，所以如何用函数丰富，底层的控制灵活的C语言写出数据库的DBF文件就要分析DBF的头部结构，通过研究测试修改，总结出可以直接使用的代码。

2 智能家居(Smart home)

“智能家居”系统是利用先进的计算机技术、通讯技术和嵌入式技术，将各种家用设备通过通信网络连接成系统。各种设备不但可以相互通讯、根据不同的状态互动运行，还可以向外提供远程控制能力，帮助家庭与外部保持信息交流畅通，从而给用户带来最大限度的高效、便利、舒适与安全。如图1所示，本文就以该系统为例来阐述远程控制的方案与实现。

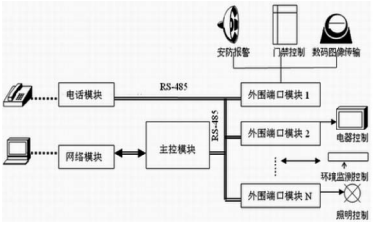

图1 系统结构示意图

Fig.1 System schematic

3 远程通信(Remote communication)

3.1 可选方案

公用电话通信网特点[2]是采用电路交换方式、信道独占、信道利用率低、传输效率高、时延小；具有通信快捷、语音明了、终端普及的优点。计算机网络特点是采用分组交换方式、信道共享、信道利用率高、传输效率低、时延大；具有费用低廉、界面直观、显示丰富的优点。二者的不同点正好可以互补，相同点是通信区域均已覆盖全球。

3.2 电话模块

电话模块具有数据处理和存储、数据通讯、语音提示、

基金项目：福建省《电子信息工程特色专业建设点》(L200802008)；福建省十二五规划《基于云计算的高校教学资源优化研究》(FJJKCG14-144)；福建省中青年教师教育科研项目《基于核函数的高维离散数据聚类算法研究与应用》(JA15586).

自动摘挂机、自动拨号等功能。具体采用8051单片机[3]作为中央控制单元，实现了语音提示及安全认证机制。主要有振铃检测、模拟摘挂机、双音频信号编解码、信号音检测、语音提示、EEPROM数据存储等子模块组成。

3.3 网络模块

实现网络通信功能的可选方案和特点有：控制芯片+网卡芯片的方案，其特点是代码编写难但网络功能灵活；控制芯片+硬件协议栈+网卡芯片的方案，其特点是控制简单但网络功能固定，如图2所示，W3100A芯片是一硬件协议栈的集成电路，芯片中安装相对简单的TCP/IP协议[4]，实现了软件硬化。

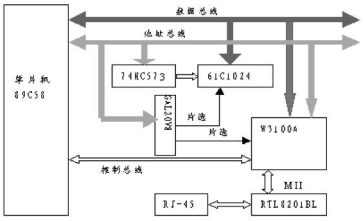

图2 网络通信模块

Fig.2 Network communication module

4 控制执行(Control execution)

外围接口模块在智能居家系统[5]负责的基本控制功能应包括：照明控制、家居安防、电器控制、环境控制等。

如图3所示，内部通信一般采用具有可扩展性和工程安装简单特点的总线技术，各个控制分支视实际情况灵活采用有线或无线技术。

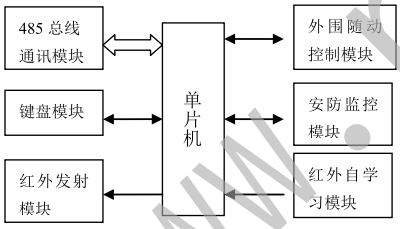

图3 外围接口模块

Fig.3 Peripheral interface module

5 C语言与数据库(C language and database)

如将这些智能终端再组成大系统时，就可能用到数据库，需要用C语言的格式读写文件。dBASE和FoxBASE都是带结构的文件，数据库文件由文件结构和数据两部分组成。文件结构采用的是二进制存储方式而数据部分则是以ASCII方式存储。库文件结构部分描述库文件的概况：建立或修改日期、数据记录个数、库文件结构长度、记录长度等。如下所示，即用C语言编写DBF文件就成了关键，经过研究调试，形成如下程序可以方便地调用。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<dos>
#include<bios.h>
#include<ctype.h>
#include<conio.h>        //此段为C标准库的头文件
struct sfox{
    unsigned char mark;      //标志
    unsigned char yy,mm,dd;   //年月日
    unsigned long recn;       //记录个数
     unsigned int strucl,recl;    //库结构长度，记录长度
    char string[20];      //保留
    }fox;                 //此段为库文件结构部分*/

struct rec{
    char sign;      //数据库的逻辑删除记号位
    char za[2];       //字段1，2位，类型不限
    char zb[2];       //字段1，2位，类型不限
    char zc[3];       //字段1，3位，类型不限
    }record;          //此段为库文件数据部分

 void write()
{
 unsigned int filehead,reccount,reclen;
 int I,start,no;
 char ch,za[2],zb[2],zc[3];
unsigned char a,b;
FILE *fpout;
If((fpout=fopen("f1.dbf","rb+"))==NULL){
          Printf("Can't open f1.dbf\n");
          }                          //打开文件
fseek(fpout,8,0);                   //定位
fscanf(fpout,"%c%c",&a,&b);     //读数据
filehead=a+b*0x100;                 //头长度
fseek(fpout,4,0);
fscanf(fpout,"%c%c",&a,&b);
reccount=a+b*0x100;                 //记录数
reclen=sizeof(struct rec);          //一条记录的长度
printf("old recount=%d\n",reccount); //显示旧记录数
printf("start new reccount=");
```

```
scanf("%d",&start);          //从第几条起加新记录
printf(" how much new reccount=");
scanf("%d",&no);             //加几条新记录
rewind(fpout);               //移动文件指针到流的开头
fread(&fox,sizeof(struct sfox),1,fout);    //按头结构
格式读入
fox.recn=start+no-1;         //新记录数
for(I=(start-1);I<(start+no-1);I++)
   {
    printf("record.za[2]=");          //提示
    scanf("%s",za);                   //键输
    strcpy(record.za,za);             //读入
    printf("record.zb[2]=");
    scanf("%s",zb);
    strcpy(record.zb,zb);
   printf("record.zc[3]=");
    scanf("%s",zc);
strcpy(record.zc,zc);
fseek(fpout,filehead+I*reclen,0);         //定位
if(fwrite(&record,sizeof(struct rec),1,fpout)==NULL){
   printf("Can't write f1.dbf\n");
   }                          //写入整条记录
}
ch=0x1a;
fwrite(&ch,1,1fpout);          //写入末尾标记
rewind(fpout);               //移动文件指针到流的开头
fwrite(&fox,sizeof(struct sfox),1,fout);    //将头结构
重新写入
fclose(fpout);               //关闭文件
 }

void read();
{
 unsigned int filehead,reccount,reclen;
 int I;
 char ch,za[2],zb[2],zc[3];
unsigned char a,b;
FILE *fpout;
If((fpout=fopen("f1.dbf","rb+"))==NULL){
        Printf("Can't open f1.dbf\n");
}
fseek(fpout,8,0);
fscanf(fpout,"%c%c",&a,&b);
filehead=a+b*0x100;
fseek(fpout,4,0);
fscanf(fpout,"%c%c",&a,&b);
reccount=a+b*0x100;
reclen=sizeof(struct rec);
        for(I=0;I<recount;I++)
           {
    fseek(fpout,filehead+I*reclen,0);
if(fread(&record,sizeof(struct rec),1,fpout)==NULL){
   printf("Can't read f1.dbf\n");
   }                          //读入整条记录
 strcpy(za,record.za);
strcpy(zb,record.zb);
strcpy(zc.record.zc);
printf("za=%s,",za);
printf("zb=%s,",zb);
printf("zc=%s,",zc);                  //显示
}
 fclose(fpout);                       //关闭文件
}
void main()                           //主函数
 {
  write();                            //调用写函数
  read();                             //调用读函数
}
```

注意：

(a)数据库的字段长度与定义的库文件数据部分对应，但字段类型可以不同。

(b)由于字符串结束符的缘故在调用“写函数”输入时只要不超过定义的位数即可，但调用“读函数”时则会出错，所以只要输入时小于定义的位数就能读写正常。

(c)此接口程序时以记录为单位进行读写的。

(d)因“写函数”中记录末尾标记的引入和新记录数的重写，记录数会随之改变。

(e)可根据需要修改程序后再放到头文件中，仅在主函数中调用即可。

(下转第48页)