·电路与控制·

# C51 与单片机系统多级菜单的模块化设计

汪高勇,宋毅恒,尚举邦

(东北电子技术研究所,辽宁 锦州 121000)

**摘 要**:介绍了一种基于单片机系统的多级菜单模块化编程方法.该方法从模块化的角度将菜单编程简单地划分为共用键盘处理和菜单模块两部分,使得菜单具有通用的模块化结构,方便了菜单的加载与卸载,增强了 C51 语言在单片机系统人机接口的设计,软件结构清晰,维护方便.

**关键词**:单片机;多级菜单;函数指针;菜单模块

**中图分类号**:TP311.11 **文献标识码**:A **文章编号**:1673-1255(2009)02-0062-05

# Multi-Lever Menu Modularization Design for C51 and Single-Chip Microcomputer System

WANG Gao-yong, SONG Yi-heng, SHANG Ju-bang

(*Northeast Research Institute of Electronics Technology, Jinzhou* 121000, *China*)

**Abstract**: A multi-level menu modularization programming method based on single-chip microcomputer system is introduced. The method classified the menu program as a menu module and a common keyboard management so as to have the menu with a common modularization structure, easy for the menu to load and unload, to enhance the C51-language design of human-computer interface in single-chip microcomputer system, and to make software structure clear, easy for maintenance.

**Key words**: single-chip microcomputer; multi-level menu; function pointer; menu module

单片机的应用中,除了进行数据运算和过程控制外,还会使用到人机接口.人机接口处理了操作者的指令输入,这些操作普遍采用了菜单方式进行命令输入,在使用液晶显示器的场合,菜单更是必需的设计.对于汇编语言而言,菜单可以使用制表法进行菜单编写,但是由于汇编语言可移植性和可读性较差,所以对于菜单的复杂性、菜单的可移植性、可维护性以及可扩充性方面,汇编语言处于弱势.使用 C 语言可以解决这方面的问题,利用 C 语言的数据结构,对菜单进行数据组织,将菜单进行模块化设计,可解决上述问题.

通常,单片机的菜单程序往往与主程序放在一起,由于菜单程序的子程序和服务比较多,会使得程序错综复杂,随着菜单的级数增多,会给代码阅读带来一定的障碍,给多级菜单的维护以及扩展带来麻烦.常用的 C51 语言菜单组成方法有状态转移法[1]、树状节点设计法[2]等常用方法,这些方法结构紧凑,构成菜单容易,但是在编程过程中,需要对菜单进行完整分析,列出节点号以及状态转移分布图,在程序编写过程中如果对程序进行功能性的修改,或者扩充,就需要重新分析节点或者状态图,对于简单的菜单系统,还不显复杂,但是对于键盘数较多,菜单级数多或菜单功能复杂的程序,就会给编程带来一定的难度,而且使用了节点或者状态序号,阅读程序时不能很清晰地了解程序运作,也给程序的管理带来了一定的难度.

对菜单进行模块化设计[3],将各菜单以独立的形式,通过打包,并考虑到资源共享和代码重用,可

**收稿日期**:2009-03-10

**作者简介**:汪高勇(1980-),男,江苏南京人,学士,研究方向为信号处理.

解决上述问题.使得开发者在软件编写过程中,只将注意力集中到局部菜单,而无需考虑状态转移和节点的全局设计.菜单的加载、卸载以及即时修改实现容易,程序结构简单,对按键的一键多意也能很好地解决.

# 1　菜单模块化设计原理

## 1.1　共用键盘处理

菜单的模块化设计主要分为共用键盘处理和菜单模块两部分.主要原理是在主程序中设置一个共用键盘处理程序,各菜单模块利用统一的键盘接口函数对共用键盘处理程序中的按键函数指针进行重定向设定,使得共用键盘处理程序能够指向各菜单模块.每次按键操作完毕,均回到主程序,这与常用的菜单编程方法类似.

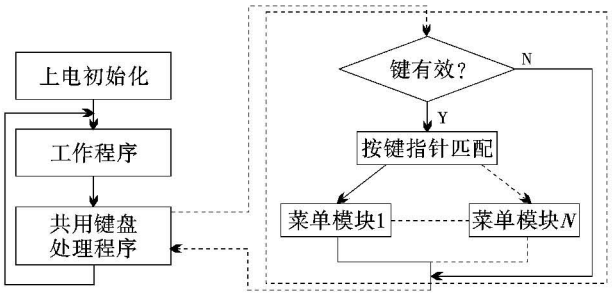


**图 1　共用键盘处理程序**

图 1 中,工作程序是软件中需要在主程序轮询执行的任务程序,用于检查系统的各种功能标识和消息并进行处理.共用键盘处理程序也算作一种任务程序,由主程序轮询执行.如图 1 虚框所示,当键标志有效时,程序对键值进行匹配,执行当前菜单模块对应的按键程序.$N$ 个菜单模块同一时刻仅有一个模块通过按键函数指针初始化得以装载.

## 1.2　菜单模块结构

菜单模块程序主要用于操作者的命令执行和信息显示,可分为初始化和本级功能程序组两部分,如图 2 所示.初始化由父菜单接口初始化、子菜单接口(子菜单存在时)初始化、屏幕初始化和按键初始化 4 部分组成.本级功能程序组则是本级菜单操作所需的命令程序,完成一些菜单或者设备的操作动作.如果该模块被调用,那么模块会进行如图 3 左侧所示的初始化步骤.屏幕初始化完成了本级菜单的界面和信息输出;按键初始化对共用键盘处理程序中的函数指针进行重定向,使其指向当前菜单模块内部的本级功能程序组;接口初始化对父菜单和子菜单接口(子菜单存在时)进行初始化.初始化完毕后,程序退出,回到主程序循环等待.当键盘有动作时,主程序中的共用键盘处理程序通过查找对应的按键函数指针,直接执行该模块本级功能程序组内的程序.执行完毕后退回到主程序继续等待或轮询执行主程序中的其他工作程序.

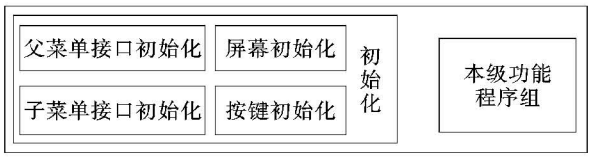


**图 2　菜单模块组成**

## 1.3　菜单模块调用关系

各菜单模块与共用键盘处理程序的相互调用关系参见图 3.菜单模块间通过父菜单与子菜单接口发生联系.在不进行菜单调用或返回动作时,共用键盘处理程序仅在当前模块状态下处理按键操作.当发生子菜单调用或父菜单返回时,本级功能程序组对子菜单或者父菜单接口进行处理,使用子菜单或者父菜单本级的初始化程序,将共用键盘处理程序中的按键函数指针重新定向到子菜单或者父菜单的本级功能程序组,实现菜单的调用与返回.对于一个多级菜单系统而言,总存在一个顶层菜单(如图 3 中的菜单 0),其下的各菜单均为其子菜单.从运行上看,它们是并列的,都是通过共用键盘处理程序进行处理,而用子菜单和父菜单接口区分了它们的上下级关系.

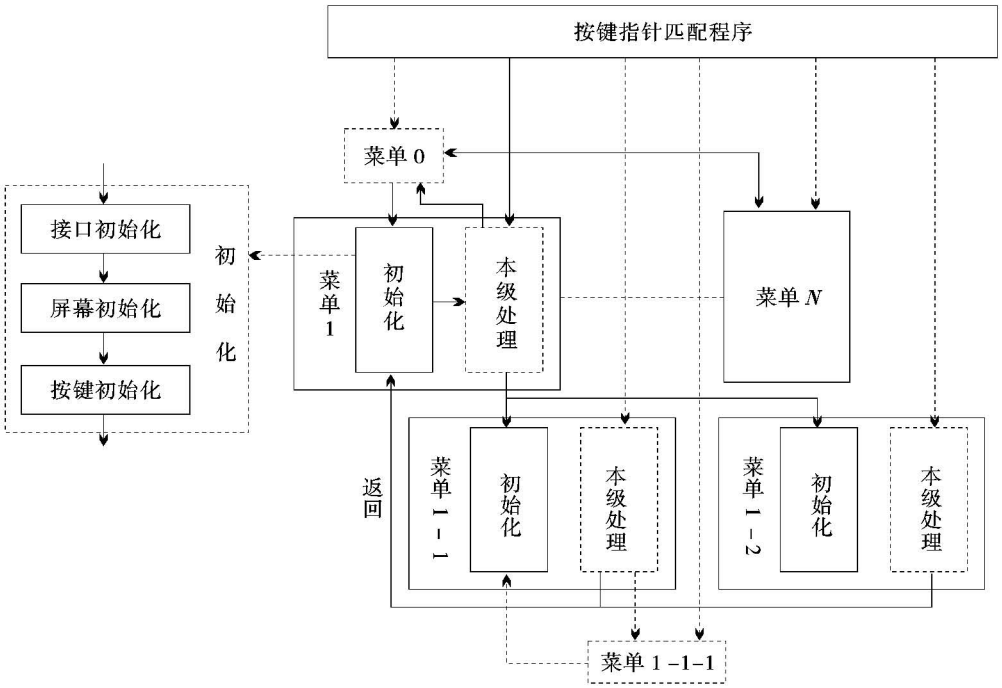


**图 3 菜单模块调用关系**

## 2 菜单模块的软件编程实现

为使键盘处理程序的结构尽量简化，通常都是复用键盘的处理程序，例如状态转移法给每个按键设定一个状态变量，通过状态变量值的不同使得同一个按键调用不同的处理程序，这需要进行状态的查找以及计算．与之类似，这里的模块化设计使用了一个键盘结构作为共用处理程序的数据结构．这里以一个具有 16 键的系统为例，说明菜单的模块化设计过程．系统中有数字键 0～9、控制键（ENTER、ESC、CLEAR）、小数点（SETUP）以及方向键（UP、DOWN）．用户通过控制键、方向键或者数字键进行菜单选择，通过数字键、小数点以及控制键进行参数设定．

该模块化结构具体包含菜单头文件、共用键盘处理程序以及菜单模块 3 部分．通过定义一些公用的变量和程序，使共用键盘处理程序以及菜单模块进行联系．菜单头文件、共用键盘处理程序可以在一个头文件中和一个 C 文件中定义．

### 2.1 建立菜单头文件

建立头文件 menu _ h.h，里面定义如下全局变量和函数．

```
#include "reg52.h"
void Key_Exe(void); //键盘处理程序声明
bit Key_FLAG; //定义键盘消息
unsigned char g_key_code; //全局键值寄存器
struct key_class{
    void (*Key_0)(); //数字键函数指针 0～9
    …
    void (*Key_9)();
    …
    void (*Key_Esc)(); //控制键函数指针
  };
typedef struct Keyclass Keystruct; //定义一个键盘结构类型
extern Keystruct Key; //定义一个全局键盘数据结构
extern void Key_Exe (uchar); //共用键盘处理程序声明
extern void Key_Rst();
//以下为各菜单模块入口声明
extern void Menu1_Main (void);
…
extern void MenuN_Main (void);
```

该结构体中的各成员变量可以看作指向不同菜单下的公用键盘向量，通过修改这些函数指针，实现了不同菜单的键盘再定义，在各菜单程序中，这些向

量可以根据使用情况作再次定义,这样可以很方便地实现一键多义.由于键值预处理程序因具体硬件而不同,所以,这里将其与菜单处理无关的宏定义一并进行了省略,不进行赘述.

## 2.2　建立共用处理程序

建立一个 Key.c 文件,里面包含下面几个程序:

```
#include  "menu_h.h"
unsigned char void_f(void)   //空函数,用于定义无效键
{;}
unsigned char Key_Rst()   //键盘向量复位程序
{
Key. Key_0= void_f;
…
Key. Key_Enter=void_f;
Key. Key_Esc=void_f;
}
void  Key_Exe(void)   //按键共用处理程序
{
if(KEY_FLAG)
{
  KEY_FLAG=0;
  switch(g_key_code)
  {
    case  KEY0:  (Key. Key_0)();
                 break;
    …
    case KEY9:  (Key. Key_9)();
                 break;
    case KEYUP:  (Key. Key_Up)();
                 break;
    …
    case KEYESC:  (Key. Key_Esc)();
                 break;
    default:          break;
    }
  }
}
```

上述程序中定义的一个 16 键函数指针结构,在程序 Key_Exe()中使用,这些键函数指针在菜单模块中重定向.Key_Exe()中的 KEY0～ KEY9 等键值由实际硬件决定,在头文件中宏定义即可.

## 2.3　建立菜单模块

假设主菜单已经定义主菜单为 menu0.c,下面建立一个子菜单文件 menu1.c 为例说明菜单模块的建立方法:

```
#include   "menu_h.h"
#define    F_MENU     Menu0_Main
//Menu0_Main 为父菜单的主程序名(入口)
#define    C_MENU     Menu?_Main
//Menu?_Main 为子菜单的主程序名(入口),"?"为编号
//修改 F_MENU 与 C_MENU 宏定义可以改变父子菜单入口
code    unsigned char Menu1_tab[]="XXXXX";   //定义本级菜单界面内容
void Menu1_Main(void)   //本级菜单入口
void Menu_Out(unsigned char *);//本级菜单输出程序
void Key_Redefine(void);   //本级菜单键盘定义
void Menu1_Main(void)   //本级菜单主程序定义
{
Key_Redefine();//键盘初始化,含接口初始化
Menu_Out(Menu1_tab);   //界面输出
}
void Menu_Out(unsigned char *csp)   //界面输出程序
{…}
void Key_Redefine()
{
Key_Rst();   //键盘复位
Key. Key_0=Work_0;
…   //根据所要使用的按键程序,重定义结构变量 Key
…   //中的成员指向本程序中的执行程序
Key. Key_9=C_MENU;   ///设置子菜单入口
Key. Key_Esc=F_MENU;   //设置父菜单入口
}
//其中"Key. Key_Esc"为返回键,若不返回,可设为其他参数或者不设置
void Work_0(void)   //本级菜单工作程序定义
{…}
…   //其余工作程序定义省略
```

按照这个结构,可以定义多个子菜单,并且使用各菜单入口函数名可以组成比较复杂的菜单操作.

## 2.4　在主程序中使用共用键盘处理程序

在主程序 main.c 中使用共用键盘处理程序 Key_Exe():

```
#include  "menu_h.h"
main()
{
  System_Ini();
  while(1)
  {
```

```
    Work1();  //工作程序 1～n
    …
    Workn();
    Key_Exe();  //共用键盘处理程序
    }
}
```

在正常运行中,程序轮询工作程序与共用键盘处理程序,当按键动作时,键盘处理程序调用相应的键值处理函数完成响应.

## 3 菜单的建立和修改

使用上述方法建立菜单,只需要给出菜单的界面参数和按键定义,按照上述菜单模块的组织方式定义本级菜单主程序(即菜单入口程序)、按键初始化程序、界面输出程序以及功能程序,并将菜单主程序在公用头文件中声明为外部函数即可.

使用时,将返回入口填入键盘处理程序中的 Key.Key_Esc 变量,如果本模块还要用到子菜单,那么可以将本级菜单主程序作为下一级菜单的返回参数,并将子菜单的主程序作为本级菜单出口.通过对头文件中的菜单模块入口声明和对应模块内父菜单入口进行修改,可对菜单进行拆卸和加载,还可将菜单扩展成多级,并且菜单返回时可以根据需要进行逐级返回或者直接返回需要到达的菜单级.

通常,可将重复的程序段进行复用设计以减少代码量.同样,也可以对菜单模块进行复用设计,在各模块中定义一个暂存变量,用于存放父菜单返回入口,父菜单调用子菜单模块时将父菜单入口作为参数传给子菜单,子菜单使用暂存变量进行保存,子菜单若再调用复用的菜单模块,按照同样的方式操作,可实现菜单的复用.但是复用会破坏菜单的树形结构,可能使得调用冲突.因此,尽量使用固定出入口进行菜单设计避免造成调用混乱.

## 4 结 论

在常用的单片机系统树形菜单设计基础上提出了一种方便程序设计和修改的模块化菜单编程方法.使用了 C51 语言进行了验证,简化了建立和修改菜单的方法,在程序编写过程中无须进行复杂的链表统计,也便于将重复的程序进行共享,使得程序编写效率较高.此外,它也是一种非阻塞式菜单设计方法[4],根据实际菜单层级关系变化可以修改菜单的层级返回顺序,在进行菜单界面修改、模块加载以及卸载时只需考虑与此菜单相关的入口和出口程序的关系而无需考虑其他的修改.该方法便于菜单程序的即时修改,使菜单具有模块化的结构,软件维护简单.

**参考文献**

[1] 王必胜,张其善.基于状态转移法的键盘程序设计[J].电子测量技术,2008(3):51－54.
[2] 李敏通,张战国.一种建立单片机应用系统菜单的新方法[J].计算机工程,2006(8):259－260,273.
[3] 朱维庆,李松平,高超.面向功能模块的通用菜单程序[J].实用程序,1993(8).
[4] 纳新,赵东风.非阻塞式 LCD 多级菜单的设计及其数据结构[J].云南民族大学学报,2007,16(4).
[5] 马忠梅.单片机的 C 语言应用程序设计[M].北京:北京航空航天大学出版社, 2003.

(上接第 53 页)

## 4 结 束 语

系统采用 LPC2294 处理器和 RTL8019AS 以太网控制器.LPC2294 有外部存储器控制器(EMC),同时支持多达 4 个单独配置的存储器组,总线宽度为 8、16、32 位可配置,和 RTL8019AS 可以很好地连接.地址数据非复用总线模式提高了数据传输速度,降低了数据传输出错机率. RTL8019AS 向 LPC2294 发送数据采用外部中断模式,保证了系统的实时性,能够很好地实现以太网功能,提升嵌入式系统的价值,具有广泛的应用价值.

**参考文献**

[1] 石风,刘成,保石.嵌入式系统设计与应用[J]. 光电技术应用,2005,20(5):44－57.
[2] 樊昌信,张甫翊.通信原理[M].北京:国防工业出版社,2001.
[3] 周立功.ARM 嵌入式系统基础教程[M].北京:北京航空航天大学出版社,2005.