为 uCOS51 增加 Shell 界面

巨龙公司系统集成开发部 杨屹 asdjf@163.com 2002/10/13

引言

自从发表《uCOS51 移植心得》以来，我收到了很多朋友们的来信，大家对公开源码表示鼓励，谢谢大家的支持！很多人对于编写自己的操作系统很感兴趣，uCOS51 是个不错的选择。它的优点是简单易懂，学习成本低，有利于向 32 位 CPU 过渡。目前，嵌入式 BBS 上的热点是：嵌入式实时多任务操作系统、单片机上网、32bitCPU（如 ARM 等）。其实通过 uCOS51 学习完全可以掌握这些热门技术的精髓，而且学习成本低廉。为此我会陆续将我在研发过程中的经验体会写出来与大家交流，共同进步。

我准备讨论以下内容：uCOS51 高效内核、OS 人机界面 SHELL 的编写、51 机开发板的硬件设计、RTL8019AS 网卡驱动程序、51TCP/IP 协议栈设计、应用协议 FTP、PPP、HTTP、SMTP、SNMP……在 51 上的实现技术、51OS 任务划分和应用程序实例、由 51 软件系统向 ARM 的移植以及其他想到的题目。欢迎大家积极参与。

注：开发板原理图、PCB 图、GAL 烧录文件、芯片手册、全部源程序可以来信索取，在整理好后会共享在网上。

讨论 2----OS 人机界面 SHELL 的编写

uCOSII 只提供了操作系统内核，用户要自己添加文件处理、人机界面、网络接口等重要部分。其中 Shell(人机界面)提供了人与机器交互的界面，是机器服务于人的体现，是系统必不可少的重要组成部分。现代的很多 OS 如 UNIX、DOS、VxWorks 都提供了友好的命令行界面。Windows 更是提供了 GUI。大部分人认识 OS 都是从这里开始的。uCOS51 同样拥有 Shell，它是我从以前写的前后台程序中移植过来的。

命令行 Shell 的工作原理比较简单，主要思路就是单片机接收用户键盘输入的字符存入命令缓冲区，并回显到屏幕，当用户按下回车键，触发软件状态机状态变迁，从输入态转移到命令解释态，然后根据用户命令调用相关子程序执行相应操作，执行完毕后重新回到输入态。

我感觉原理很好掌握，程序也不长，但是细节部分要反复调试多次才能稳定工作。比如：命令行左右边界的保护、退格键的处理、词表的设计等等。

Shell 程序由词表、取词子程序、状态机框架程序(输入回显和命令解释执行)、命令相关子程序组成(详见源程序清单)。

词表结构如程序清单所示，由词数目，左括号数，右括号数，每个词的具体信息（长度，字符串）构成。左右括号数用于括号匹配检查；词数目用于程序循环；词的具体信息作为解释/执行程序的输入参数。

取词子程序从命令行语句中提取单词并存入词表同时进行匹配检查和词法分析。默认字符为：0-9、a-z、A-Z、'.'；定界符为：空格、逗号，左/右括号。建议用户补充默认字符集(? / \ -)以便实现更灵活的语法。注意：现在版本的 Shell 只检查左右括号数量的匹配，无优先级和语法含义。

输入回显程序循环检查用户键盘输入。如果输入回车，程序状态转入解释执行态；如果

输入退格(8)则回显退格、空格、退格，模拟删除字符，同时输入缓冲区清除相应字节，清除前先检查左边界是否越界。如越界则鸣响报警且不执行清除操作；其他字符输入直接存入输入缓冲区并回显，此前检查右边界是否溢出，如果溢出则鸣响报警且抛弃刚输入的字符。

命令解释程序调用取词子程序分析用户命令行输入，根据词表第一个单词在散转表中的位置调用相应执行子程序处理命令，如果散转表中无此单词，则打印“Bad command!”。取词子程序返回错误指示时也打印此句。

命令解释程序向相应的命令相关子程序传入词表指针，具体执行由用户自行决定。由命令相关子程序返回后重新回到命令输入态，完成一次输入执行全过程。此过程周而复始地循环执行。

Shell 界面的命令按功能分为以下几组：

1。操作系统相关命令：

查看就绪任务 lt / 中止任务 kill / 恢复任务执行 call / CPU 利用率 usage / 版本查询 ver / 查某个任务信息(TCB、堆栈内容)lt

查看切换次数和时间 lts

2。网络相关命令：

显示配置 MAC 地址 macadr / 显示配置主机 IP 地址 host / 显示配置子网掩码 mask / 显示配置缺省网关 gateway

显示网络配置总情况 lc / 连通测试命令 ping / 用户数据报发送命令 udp / telnet 命令 tel / 相关应用命令**

显示 ARP 高速缓冲区地址对 ls / 显示发送缓冲区信息 lti

3。屏幕显示相关命令：

清屏 clr / 帮助 help / 功能键 F3、F7 处理 / 组合键 Ctrl+C、Ctrl+B 处理

4。外设(闪盘 X5045 和 I/O 口)相关命令：

读闪盘 rdx / 读 I/O 口 rdp / 写闪盘 wdx

5。安全相关命令：

身份认证密码权限 usr、pass

6。应用相关命令：

用户自行定义

7。调试相关命令：

单步、断点、运行、寄存器等命令，类似 68K 的 TUTOR 和 ARM 的驻留监控程序 Angel。

怎么样，像不像 VxWorks 的命令行 Shell！

用户命令大小写不敏感，程序将命令字符串统一成小写形式。程序中各种参数(如：最大词长度、词数量……)定义成宏放在一个头文件中，随时可修改配置，很方便。Shell 作为一个任务工作于内核之外，占用一个任务号。

源程序：

词表

```
typedef struct{
    int Num;
    int LeftCurveNum,RightCurveNum;
    struct{
        int Length;
        unsigned char Str[MaxLenWord+1];    /*for '\0'*/
    } wt[MaxLenWordTable];
} WORDTABLE;
```

取词

```
bit GetWord(unsigned char *ComBuf,WORDTABLE *WordTable)
{
    int i=0;    /*ComBuf String pointer*/
    int j=0;    /*Length of Word */
    int k=-1;  /*The number of WordTable*/
    int StrFlag=0;  /*There is "0-9/a-z/A-Z" before " ,()"*/
    int SentenceEndFlag=0; /*Sentence end*/
    char ch;

    WordTable->Num=0;
    WordTable->LeftCurveNum=0;
    WordTable->RightCurveNum=0;

    ch=ComBuf[0];
    while(!SentenceEndFlag&&i<MaxLenComBuf){
        if((ch>='0'&&ch<='9')||(ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')||(ch=='.')){
            if(StrFlag==0){
                StrFlag=1;k=k+1;j=0;
                if(k>=MaxLenWordTable) return 0;
                WordTable->wt[k].Str[j]=ch;
                WordTable->Num=k+1;
            }
            else{
                j=j+1;
                if(j>=MaxLenWord) return 0;
                WordTable->wt[k].Str[j]=ch;
            }
        }
        else if(ch==' '||ch==','||ch=='('||ch==')'||ch=='\0'){
            if(ch=='(') WordTable->LeftCurveNum++;
            if(ch==')') WordTable->RightCurveNum++;
```

```
            if(StrFlag==1){
                StrFlag=0;j=j+1;
                WordTable->wt[k].Str[j]='\0';
                WordTable->wt[k].Length=j;
            }
            if(ch=='\0') SentenceEndFlag=1;
        }
        else{
            return 0;
        }
        i=i+1;
        ch=ComBuf[i];
    }
    if(i<MaxLenComBuf||ComBuf[MaxLenComBuf]=='\0'){
        if(WordTable->LeftCurveNum==WordTable->RightCurveNum) return 1;
        else return 0;
    }
    else{
        return 0;
    }
}
```

输入回显和命令解释执行

```
void yyshell(void *yydata) reentrant
{
    yydata=yydata;
    clrscr();
    PrintStr("\t\t***********************************************\n");
    PrintStr("\t\t*              Welcom to use this program              *\n");
    PrintStr("\t\t*                          Author:YangYi 20020715      *\n");
    PrintStr("\t\t***********************************************\n\n\n");

    /*Login & Password*/

    PrintStr("% ");
    while(!ShellEnd){

        switch(State){
            case StatInputCom:{
                if(yygetch(&ch)){
                    if(ch==13)      /*Enter return key*/
                    {
                        PrintStr("\n");
                        ComBuf[i+1]='\0';
```

```
                        if(i+1==0) PrintStr("% ");
                        else
                            State=StatExeCom;
                    }
                    else{
                        i=i+1;
                        if((i>=MaxLenComBuf)&&(ch!=8)){
                            PrintChar(7);
                            i=MaxLenComBuf-1;
                        }
                        else{
                            if(ch==8){
                                i=i-2;
                                if(i<-1) {i=-1;PrintChar(7);}
                                else{
                                    PrintChar(8);
                                    PrintChar(' ');
                                    PrintChar(8);
                                }
                            }
                            else{
                                PrintChar(ch);
                                ComBuf[i]=ch;
                            }
                        }
                    }
                    break;
                }
                else{
                    //OSTimeDly(10);
                    break;
                }
            }
            case StatExeCom:{
                if(GetWord(ComBuf,&WordTable)==1&&WordTable.Num!=0){
                    yystrlwr(WordTable.wt[0].Str);
                    for(tem=0;tem<MaxComNum&&!ComMatchFlag;tem++)
                        if(yystrcmp(WordTable.wt[0].Str,ComTable[tem])==0)
ComMatchFlag=1;
                    if(ComMatchFlag){
                        tem--;
                        switch(tem){
                            case 0:{DisplayTask(&WordTable);break;}
                            case 1:{Kill(&WordTable);break;}
```

```
                                case 2:{PingCommand(&WordTable);break;}
                                case 3:{UDPCommand(&WordTable);break;}
                                case 4:{CfgHost(&WordTable);break;}
                                case 5:{CfgMask(&WordTable);break;}
                                case 6:{CfgGateway(&WordTable);break;}

                                case 7:{
                                    //ShellEnd=1;
                                    PrintStr("\n\tThis Command is limited!\n\n");
                                    break;
                                }
                                case 8:{PrintConfig(&WordTable);break;}
                                case 9:{clrscr();break;}
                                case 10:{DisplayHelpMenu(&WordTable);break;}
                            }
                        }
                        else
                            PrintStr("      Bad command!\n\n");
                    }
                    else{
                        if(WordTable.Num) PrintStr("      Bad command!\n\n");
                    }

                    ComMatchFlag=0;
                    State=StatInputCom;
                    if(ShellEnd) {PrintStr("\n\n");}
                    else PrintStr("% ");
                    i=-1;
                    break;
                }
                default:{
                    //ShellEnd=1;
                    PrintStr("System fatal error!\n");
                    PrintChar(7);PrintChar(7);PrintChar(7);
                }
            }
        }
    }
```

命令使用说明：

一、关于 93LC66 的命令

1.rd66 [起址] [显示行数]

例如：rd66 0x100 3　　显示从 0x100 开始的数据，显示 3 行（每行 16 字节）。

注：显示行数最大为 ROMSIZE/16；省略起址和显示行数，则缺省显示 0x00 开始的一行，省略显示行数，则缺省显示 1 行。另有一 rdb66 指令，格式与此同，只是它是按字节显示的。

2.wd66 地址 [x] 数据 0-n

例如：wd66 all 0xad　将 0xad 写入 93LC66 全部存储单元。

wd66 0xaa 0xff 将 0xff 写入 0xaa 存储单元。

wd66 0xcd x aa bb cc dd 将十六进制数 AA、BB、CC、DD 写入 0xcd 单元。

wd66 0xcd 12 34 56 78　　将十进制数 12、34、56、78 写入 0xcd 单元。

注：地址和数据不可省略;标记 x 用于数据进制的指定。

3.erase66 地址

例如：erase66 all　　擦除 93LC66 全部存储单元。

erase66 0x20　　擦除 0x20 存储单元。

注：地址不可省略。全擦命令要求确认一次（按 y/Y 确认）。

二、关于 X5045 的命令

1.rxr

例如：rxr　　读 X5045 状态寄存器。

注：状态寄存器格式 X　X　WD1　WD0　BL1　BL0　WEL　WIP

WD1	WD0		BL1	BL0	
0	0	1.4s	0	0	None
0	1	600ms	0	1	$180-$1FF
1	0	200ms	1	0	$100-$1FF
1	1	Disable	1	1	$000-$1FF

WIP=0 空闲；WIP=1 正在写；WEL=0 不可写；WEL=1 可写。

2.wxr

例如：wxr　　写 X5045 状态寄存器。

注：见 rxr 注。

3.rdx [起址] [显示行数]

例如:：rdx 0x100 3　　显示从 0x100 开始的数据，显示 3 行（每行 16 字节）。

注：显示行数最大为 ROMSIZE/16；省略起址和显示行数，则缺省显示 0x00 开始的一行，省略显示行数，则缺省显示 1 行。ROMSIZE 按字节为单位计算。

4.wdx 地址 [x] 数据 0-n

例如：wdx 0xaa 0xff 将 0xff 写入 0xaa 存储单元。

wdx 0xcd x aa bb cc dd 将十六进制数 AA、BB、CC、DD 写入 0xcd 单元。

wdx 0xcd 12 34 56 78　　将十进制数 12、34、56、78 写入 0xcd 单元。

注：地址和数据不可省略;标记 x 用于数据进制的指定。

三、关于端口的命令

1.rdp

例如：rdp　　读端口值。
注：背面 JACK …BA3 BA2 BA1 BA0 POW2 POW1
　　正面 ISIN LINKOK TXSPEED … JACK

2.wrbs
例如：wrbs　写板选（0-15）。
注：见 rdp 注。

四、关于 shell 的命令
1.clr
例如：clr　　清屏幕。

2.exit
例如：exit　退出 Shell。
注：此命令受限制，此处不响应。

3.help
例如：help　屏幕打印帮助菜单。

五、关于 TCPIP 的命令
1.lc
例如：lc　　显示 TCPIP 配置信息（MAC 地址、主机地址、子网掩码、网关地址、SNMP 主机地址）。

2.ls
例如：ls　　显示 ARP 缓存信息（寿命、MAC 地址、IP 地址）。

3.lt
例如：lt　　显示发包缓存信息（目的 IP 地址、重发次数）。

4.host
例如：host　显示主机 IP 地址。
　　host 172.18.92.86 更改 IP 地址为 172.18.92.86。

5.mask
例如：mask　显示子网掩码。
　　mask 255.255.0.0 更改子网掩码为 255.255.0.0。

6.gateway
例如：gateway　显示网关地址。
　　gateway　172.18.80.1 更改网关地址为 172.18.80.1。

7.snmphost
例如：snmphost　显示 SNMP 主机地址。

snmphost　172.18.92.66 更改 SNMP 主机地址为 172.18.92.66。

8.macadr

例如：macadr　显示 MAC 地址。

macadr　0000 1234 5678 更改 MAC 地址为 0000 1234 5678。

9.ping

例如：ping 172.18.92.87　检查目的主机连通情况。

10.udp

例如：udp 172.18.92.87 Hello. 向 172.18.92.87 发送 UDP 包，包内数据为字符串。

X5045 数据结构

地址类型	对应全局变量	位置	大小
MAC	my_ethernet_address	0-5	6
IP	my_ip_address	6-9	4
MASK	mask_ip_address	10-13	4
GateWay IP	gateway_ip_address	14-17	4
SNMP IP	SNMP_ip_address	18-21	4