# C51 的编程规范简介

**（ 资料来源：转载）**

编程首要是要考虑程序的可行性，然后是可读性、可移植性、健壮性以及可测试性。这是总则。但是很多人忽略了可读性、可移植性和健壮性(可调试的方法可能歌不相同)，这是不对的。

1.当项目比较大时，最好分模块编程，一个模块一个程序，很方便修改，也便于重用和便于阅读。

2.每个文件的开头应该写明这个文件是哪个项目里的哪个模块，是在什么编译环境下编译的，编程者(/修改者)和编程日期，值得注意的是一定不要忘了编程日期，因为以后你再看文件时，会知道大概是什么时候编写的，有些什么功能，并且可能知道类似模块之间的差异(有时同一模块所用的资源不同，和单片机相连的方法也不同，或者只是在原有的模块上加以改进)。

3.一个C源文件配置一个h头文件或者整个项目的C文件配置一个h头文件，我自己采用整个项目的C文件配置一个h头文件的方法，并且使用#ifndef/#define/#endif 的宏来防止重复定义，方便各模块之间相互调用。

4.一些常量(如圆周率 PI)或者常需要在调试时修改的参数最好用#define 定义，但要注意宏定义只是简单的替换，因此有些括号不可少。

5.不要轻易调用某些库函数，因为有些库函数代码很长(我是反对使用 printf 之类的库函数的，但是是一家之言，并不勉强各位)。

6.书写代码时要注意括号对齐，固定缩进，一个{}各占一行，我本人采用采用所进 4 个字符，应该还是比较合适的，if/for/while/do 等语句各占一行，执行语句不得紧跟其后，无论执行语句多少都要加{},千万不要写成如下格式：

```
for(i=0;i<100;i++){fun1();fun2();}

for(i=0;i<100;i++){

fun1();

fun2();

}
```

而应该写成：

```
for(i=0;i<100;i++)
```

```
{
fun1();
fun2();
}
```

7.一行只实现一个功能，比如：

```
a=2;b=3;c=4;宜改成：
a=2;
b=3;
c=4;
```

8.重要难懂的代码要写注释，每个函数要写注释，每个全局变量要写注释，一些局部变量也要写注释。注释写在代码的上方或者右方，千万不要写在下方(相信没有人写在左方吧:))。

9.对各运算符的优先级右所了解，记不得没关系，加括号就是，千万不要自作聪明说自己记得很牢。

10.不管有没有无效分支，switch 函数一定要 defaut 这个分支。一来让阅读者知道程序员并没有遗忘 default,并且防止程序运行过程中出现的意外(健壮性)。

11.变量和函数的命名最好能做到望文生义。不要命名什么 x，y，z，a，sdrf 之类的名字。

12.函数的参数和返回值没有的话最好使用 void。

13.goto 语句：从汇编转型成 C 的人很喜欢用 goto，但 goto 是 C 语言的大忌，但是老实说，程序出错是程序员自己造成的，不是 goto 的过错；本人只推荐一种情况下使用 goto 语句，即从多层循环体中跳出。

14.指针是 C 语言的精华，但是在 C51 中我个人认为少用为妙，一来有时反而要花费多的空间，还有在对片外数据进行操作时会出错(可能是时序的问题)。

15.一些常数和表格之类的应该放到 code 去中以节省 RAM。

16.程序编完编译看有多少 code 多少 data，注意不要使堆栈为难。

17.程序应该要能方便的进行测试，其实这也与编程的思维有关；一般有三种：一种是自上而下先整体再局部；一种是自下而上先局部再整体；还有一种是结合两者往中间凑。我的做法是现大概规划一下整个编程，然后一个模块模块独立编程，每个模块调试成功再拼凑在一块调试。我建议：如果程序不大，可以直接用一个文件直接编，如果程序很大，宜采用自上而下的方式，但更多的是那种处在中间的情况，宜采用自下而上或者结合的方式。

//以下是《模块》或《文件》注释内容：

//////////////////////////////////////////////////////////////////////////

//公司名称：

//模 块 名：

//创 建 者：

//修 改 者：

//功能描述：

//其他说明：

//版本：

//////////////////////////////////////////////////////////////////////////

//以下是《函数》注释内容：

//////////////////////////////////////////////////////////////////////////

//函 数 名：

//功能描述：

//函数说明：

//调用函数：

//全局变量：

//输入：

//返回：

//设 计 者：

//修 改 者：

//版本：

////////////////////////////////////////////////////////////////////////

作为一门工具，最终的目的就是实现功能。在满足这个前提条件下，我们希望我们的程序能很容易地被别人读懂，或者能够很容易地读懂别人的程序，在团体合作开发中就能起到事半功倍之效。在网上请求帮助时，如能以规范的写法贴出程序，网友会比较容易地明白你的问题，则会比较快的得到网友的帮助，否则让人看上半天也不明所以然，这样就达不到预期的效果了。因此，为了便于源程序的交流，减少合作开发中的障碍，希望大家能够探讨一下 C51 的编程规范。把各人认为好的建议提出来，然后做一个总结，作为一种大家一致认同的规范，我认为将会是一件很有意义的事。我先提出一些自己的想法，以此抛砖引玉。

一、注释

1，采用中文；

2，开始的注释：

文件(模块)注释内容：

公司名称、版权、作者名称、修改时间、模块功能、背景介绍等,复杂的算法需要加上流程说明;

函数开头的注释内容：

函数名称、功能、说明输入、返回、函数描述、流程处理、全局变量、调用样例等,复杂的函数需要加上变量用途说明;

3、程序中的注释内容：

修改时间和作者、方便理解的注释等。注释内容应简炼、清楚、明了，一目了然的语句不加注释。

二、命名：

命名必须具有一定的实际意义。

1、常量的命名：全部用大写。

2、变量的命名：

变量名加前缀，前缀反映变量的数据类型，用小写，反映变量意义的第一个字母大写，其他小写。

其中变量数据类型：

unsigned char 前缀 ucsigned char 前缀 sc

unsigned int 前缀 uisigned int 前缀 si

unsigned long 前缀 ulsigned long 前缀 sl

bit 前缀 b 指针前缀 p

例：ucReceivData 接收数据

3、结构体命名：

4、函数的命名：

函数名首字大写，若包含有两个单词的每个单词首字母大写。

函数原型说明包括：引用外来函数及内部函数，外部引用必须在右侧注明函数来源：模块名及文件名，内部函数，只要注释其定义文件名;

三、编辑风格

1、缩进：缩进以 Tab 为单位，一个 Tab 为四个空格大小。预处理语句、全局数据、函数原型、标题、附加说明、函数说明、标号等均顶格书写。语句块的“{”“}”配对对齐，并与其前一行对齐;

2、空格：数据和函数在其类型，修饰名称之间适当空格并据情况对齐。关键字原则上空一格，如：

if ( ... ) 等，运算符的空格规定如下：“->”、“[”、“]”、“++”、“--”、“~”、“!”、“+”、“-”（指正负号），“&”（取址或引用）、“*”（指使用指针时）等几个运算符两边不空格（其中单目运算符系指与操作数相连的一边），其它运算符（包括大多数二目运算符和三目运算符“?:”两边均空一格，“(”、“)”运算符在其内侧空一格，在作函数定义时还可据情况多空或不空格来对齐，但在函数实现时可以不用。“,”运算符只在其后空一格，

需对齐时也可不空或多空格，对语句行后加的注释应用适当空格与语句隔开并尽可能对齐。

3、对齐：原则上关系密切的行应对齐，对齐包括类型、修饰、名称、参数等各部分对齐。另每一行的长度不应超过屏幕太多，必要时适当换行，换行时尽可能在“,”处或运算符处，换行后最好以运算符打头，并且以下各行均以该语句首行缩进，但该语句仍以首行的缩进为准，即如其下一行为“{”应与首行对齐。

4、空行：程序文件结构各部分之间空两行，若不必要也可只空一行，各函数实现之间一般空两行

5、修改：版本封存以后的修改一定要将老语句用封闭，不能自行删除或修改,并要在文件及函数的修改记录中加以记录。

6、形参：在定义函数时，在函数名后面括号中直接进行形式参数说明，不再另行说明。