

目录

- 一 [标准文件操作](#)
- 二 [文件的使用](#)
- 三 [程序的版式](#)
- 四 [程序的注释](#)
- 五 [命名规则](#)
- 六 [表达式和基本语句](#)
- 七 [函数设计](#)
- 八 [软件使用规范](#)
- 附录一 [软件开发信息传递表（函数信息）](#)
- 附录二 [软件标准文件操作记录表](#)
- 附录三 [软件编程规范审查表](#)
- 附录四 [执行情况](#)
- 附录五 [软件设计流程及规范](#)

一、标准文件操作

- 1.1 安装保密软件 BestCrypt，安装目录不限。
- 1.2 在 BestCrypt Container 下创建以下几个文件夹（见表 1-1）

目录	功能	备注
T:\C51STD	存放各种标准文件夹	强制
T:\C51STD\Demo	存放示例程序文件	建议
T:\C51STD\Source	存放库源程序文件	建议
T:\C51STD\Head	存放头文件	强制
T:\C51STD\Lib	存放库文件	强制
T:\C51STD\History	存放废止不用的文件	强制
T:\C51STD\《软件标准文件修改记录表.doc》	记录各种标准文件的更改内容	强制

表 1-1

- 1.3 标准文件的文件变动仅能由研发部经理指定的专人负责，在 T:\C51STD\History 下自行建立文件夹，并且将废弃不用的文件均应存放于建立的文件夹中，文件夹的名字为修改日期。
- 1.4 标准头文件以及库文件需研发部全体讨论后建立为标准共用文件，建立后改为只读属性。
- 1.5 标准文件均应在文件名后添加版本信息，如***_V100.h，***_V100.lib。
- 1.6 标准文件操作后应提交以下文件
 - 1.61 源文件，库文件，头文件
 - 1.62 软件流程图
 - 1.63 《软件标准文件修改记录表》

二、文件的使用

- 2.1 为了防止头文件被重复引用，应当用ifndef/define/endif 结构产生预处理块。
- 2.2 用 #include <filename.h> 格式来引用标准库的头文件（编译器将从标准库目录开始搜索）。
- 2.3 用 #include "filename.h" 格式来引用非标准库的头文件（编译器将从用户的工作目录开始搜索）
- 2.4 头文件中只存放“声明”而不存放“定义”。
- 2.5 头文件不提倡使用全局变量，尽量不要在头文件中出现extern int value 这类声明。

三、程序的版式

- 3.1 程序块要采用缩进风格编写，缩进使用Tab键，Tab键约定为4个空格，不允许使用空格代替TAB键。
- 3.2 一行代码只做一件事情，只写一条语句。这样的代码容易阅读，并且方便于写注释。

示例：如下例子不符合规范。

```
rect.length = 0; rect.width = 0;
```

应如下书写

```
rect.length = 0;

rect.width = 0;
```

- 3.3 if、for、while、do 等语句自占一行，执行语句不得紧跟其后。不论执行语句有多少都要加{}。这样可以防止书写失误。

示例：如下例子不符合规范。

```
if (pUserCR == NULL) return;
```

应如下书写：

```
if (pUserCR == NULL)
{
    return;
}
```

3.4 在每个结构声明之后、每个函数定义结束之后都要加“//=====”;

3.5 在一个函数体内，逻辑上密切相关的语句之间不加空行，其它地方应加空行分隔，相对独立的大功能描述语句之间加“//-----”

3.6 ‘,’之后要留空格；如果‘;’不是一行的结束符号，其后要留空格。

```
示例：  
  
Function(x, y, z)  
  
for(initialization; condition; update)。
```

3.7 程序的分界符‘{’和‘}’应独占一行并且位于同一列，同时与引用它们的语句左对齐。

3.8 { } 之间的代码块在‘{’左边列对齐。

3.9 较长的语句(>80 字符)要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在新行之首，划分出的新行要进行适当的缩进，使排版整齐，语句可读。

```
示例：  
  
perm_count_msg.head.len = NO7_TO_STAT_PERM_COUNT_LEN  
  
                        + STAT_SIZE_PER_FRAM * sizeof( _UL );  
  
  
act_task_table[frame_id * STAT_TASK_CHECK_NUMBER + index].occupied  
    = stat_poi[index].occupied;  
act_task_table[taskno].duration_true_or_false  
    = SYS_get_sccp_statistic_state( stat_item );  
report_or_not_flag = ((taskno < MAX_ACT_TASK_NUMBER)  
  
                    && (n7stat_stat_item_valid (stat_item))  
  
                    && (act_task_table[taskno].result_data != 0));
```

四、程序的注释

4.1 注释行数不得低于代码行数的60%。

4.2 尽量把注释放在代码行的最后，如果一行当中代码过长影响注释查看的，把注释放在代码行的上一行，不可放在下方。

4.3 边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。不再有用的注释要删除。

4.4 尽量避免在注释中使用缩写，特别是不常用缩写。

4.5 当代码比较长，特别是有多重嵌套时，应当在一些段落的结束处加注释，便于阅读。

4.6 文件（如头文件.h文件、.inc文件、源文件等）头部应进行注释，注释必须列出：版权说明、版本号、生成日期、作者、内容、功能、与其它文件的关系、修改日志等，头文件的注释中还应函数功能简要说明；版本改变时保留上一版本的文件注释。

```
示例：

/*****

Copyright (C), 2003-2009, Winpark Electronics Co., Ltd.

File name:      // 文件名

Author:      Version:      Date: // 作者、版本及完成日期

Description:    // 用于详细说明此程序文件完成的主要功能，与其他模块
                // 或函数的接口，输出值、取值范围、含义及参数间的控
                // 制、顺序、独立或依赖等关系

Function List:  // 主要函数列表，每条记录应包括函数名及功能简要说明

1.    ....

Change List:    //更改列表，用于描述与上一版本相比更改了哪些内容

*****/
```

4.7 函数头部应进行注释，列出：函数的目的/功能、输入参数、输出参数、返回值、等。

```
示例：

/*****

Function:      // 函数名称

Description:    // 函数功能、性能等的描述

Calls:         // 被本函数调用的函数清单

Called By:     // 调用本函数的函数清单

Input:         // 输入参数说明，包括每个参数的作用、取值说明及参数间关系。

Output:        // 对输出参数的说明。

Return:        // 函数返回值的说明

Others:        // 其它说明

*****/
```

五、命名规则

5.1 程序中不要出现标识符完全相同的局部变量和全局变量，尽管两者的作用域不同而不会发生语法错误，但会使人误解。

5.2 尽量避免名字中出现数字编号，如Value1, Value2 等，除非逻辑上的确需要编号。

5.3 使用英文缩写命名，避免名称的拼音写法。

5.4 宏定义命名规则：M_(大写的定义名称)

5.5 typedef定义命名规则：TP_(大写的定义名称)

5.6 常量命名规则：C_(大写的定义名称)

5.7 函数名称命名规则：全部小写

5.8 变量定义命名规则：(b, c, i, l, f, s, u)_(小写的定义名称)

示例：int lsb_Width (见表1-2)

变量范围	数据符号	数据类型
局部 (local) :l	有符号 (signed) :s	位 (bit) :b
全局 (global) :g		字符 (char) :c
		整形 (integer) :i
		长整形 (long) :l
		结构 (struct) :s
		联合 (union) :u

表1-2

六、表达式和基本语句

- 6.1 如果代码行中的运算符比较多，用括号确定表达式的操作顺序，避免使用默认的优先级。
- 6.2 在多重循环中，如果有可能，应当将最长的循环放在最内层，最短的循环放在最外层，以减少CPU 跨切循环层的次数
- 6.3 不可在for 循环体内修改循环变量，防止for 循环失去控制。
- 6.4 不要忘记最后那个default 分支。即使程序真的不需要default 处理，也应该保留语句 default : break;
- 6.5 需要对外公开的常量放在头文件中，不需要对外公开的常量放在定义文件的头部。为便于管理，可以把不同模块的常量集中存放在一个公共的头文件中。
- 6.6 函数内超过两处相同的常量必须使用宏定义。
- 6.7 如果某一常量与其它常量密切相关，应在定义中包含这种关系，而不应给出一些孤立的值。
- 6.8 宏定义必须加括号。
- 6.9 用括号明确表达式的操作顺序，避免使用默认优先级。
- 6.10 存在隐式转换的语句必须强制转换。


七、函数设计

- 7.1 参数的书写要完整，不要贪图省事只写参数的类型而省略参数名字。如果函数没有参数，则用 void 填充。
- 7.2 必要时在函数体的“入口处”，对参数的有效性进行检查。
- 7.3 必要时在函数体的“出口处”，对 return 语句的正确性和效率进行检查。
- 7.4 避免函数有太多的参数，参数个数尽量控制在 5 个以内。
- 7.5 函数的功能要单一，不要设计多用途的函数。
- 7.6 养成“使用调试器逐步跟踪程序”的习惯，只有这样才能发现问题的本质。
- 7.7 函数完成后必须检查所用使用变量的同步问题。


八、软件使用规范

- 8.1 编译前应检查自己的 Keil 版本，保证版本应和技术部指定版本保持一致
- 8.2 统一使用 UltraEdit 软件进行代码编辑。

附录一：

软件开发信息传递表（函数信息）				
输入信息		输出信息		
函数（宏）名称		影响到的资源	Port	
			SFR	
			INT	
函数功能		RAM 使用	动态	
			静态	
			全局	
传递参数		使用的标准文件	Head	
			Lib	
返回参数		变量同步要求		
硬件资源				
全局变量				
初始化信息				
验收结果				
 WINPARK ELECTRONICS www.cnwinpark.com		文件编号	CXHZ0000	
		输出/日期	某某某	2009-7-14
		验证/日期	某某某	2009-7-14
		批准/日期	某某某	2009-7-14

附录二：

软件标准文件操作记录表			
文件包名称		提交文件	<input type="checkbox"/> 项目文件 <input type="checkbox"/> 源文件 <input type="checkbox"/> 库文件 <input type="checkbox"/> 头文件 <input type="checkbox"/> 流程图 <input type="checkbox"/> 其他
操作类型	<input type="checkbox"/> 新建 <input type="checkbox"/> 删除 <input type="checkbox"/> 修改	向下兼容	<input type="checkbox"/> 是 <input type="checkbox"/> 否 <input type="checkbox"/> 无
新建/修改纪要	<input type="checkbox"/> 程序 Bug <input type="checkbox"/> 功能改进 <input type="checkbox"/> 功能添加		
	功能描述		
验证过程	新功能是否实现 <input type="checkbox"/> 是 <input type="checkbox"/> 否 是否影响其他功能 <input type="checkbox"/> 是 <input type="checkbox"/> 否 是否有异常情况 <input type="checkbox"/> 是 <input type="checkbox"/> 否		
	验证记录和结论：		
 WINPARK ELECTRONICS www.cnwinpark.com		文件编号	CXBZ0000
		输出/日期	某某某 2009-7-14
		验证/日期	某某某 2009-7-14
		批准/日期	某某某 2009-7-14

附录三：

软件编程规范审查表			
检查人		检查日期	
检查对象		审查结果	通过□ 不通过□
审查内容			
说明			

附录四：

序号	总 则 条 款	执行情况	说明
一：标准文件操作			
1.1	安装保密软件BestCrypt，安装目录不限。	是[] 否[] 免[]	
1.2	在BestCrypt Container下创建以下几个文件夹	是[] 否[] 免[]	
1.3	标准文件的文件变动仅能由研发部经理指定的专人负责，在 T:\C51STD\History 下自行建立文件夹，并且将废弃不用的文件均应存放于建立的文件夹中，文件夹的名字为修改日期。	是[] 否[] 免[]	
1.4	标准头文件以及库文件需研发部全体讨论后建立为标准共用文件，建立后改为只读属性	是[] 否[] 免[]	
1.5	标准文件均应在文件名后添加版本信息，如***_V100.h，***_V100.lib	是[] 否[] 免[]	
1.6	标准文件操作后应提交以下文件：1.61 源文件，库文件，头文件、1.62 软件流程图、1.63 《软件标准文件修改记录表》	是[] 否[] 免[]	
二：文件的使用			
2.1	为了防止头文件被重复引用，应当用ifndef/define/endif 结构产生预处理块。	是[] 否[] 免[]	
2.2	用 #include <filename.h> 格式来引用标准库的头文件（编译器将从标准库目录开始搜索）	是[] 否[] 免[]	
2.3	用 #include "filename.h" 格式来引用非标准库的头文件（编译器将从用户的工作目录开始搜索）	是[] 否[] 免[]	
2.4	头文件中只存放“声明”而不存放“定义”。	是[] 否[] 免[]	
2.5	头文件不提倡使用全局变量，尽量不要在头文件中出现extern int value 这类声明	是[] 否[] 免[]	
三：程序的版式			
3.1	程序块要采用缩进风格编写，缩进使用Tab键，Tab约定为4个空格，不允许使用空格代替TAB键	是[] 否[] 免[]	
3.2	一行代码只做一件事情，只写一条语句。这样的代码容易阅读，并且方便于写注释	是[] 否[] 免[]	
3.3	if、for、while、do 等语句自占一行，执行语句不得紧跟其后。不论执行语句有多少都要加{}。这样可以防止书写失误。	是[] 否[] 免[]	
3.4	在每个结构声明之后、每个函数定义结束之后都要加“//=====”	是[] 否[] 免[]	
3.5	在一个函数体内，逻辑上密切相关的语句之间不加空行，其它地方应加空行分隔，相对独立的大功能描述语句之间加“//-----”	是[] 否[] 免[]	

序号	总 则 条 款	执行情况	说明
3.6	‘,’ 之后要留空格; 如果 ‘;’ 不是一行的结束符号, 其后要留空格	是[] 否[] 免[]	
3.7	程序的分界符 ‘{’ 和 ‘}’ 应独占一行并且位于同一列, 同时与引用它们的语句左对齐	是[] 否[] 免[]	
3.8	{ } 之中的代码块在 ‘{’ 左边列对齐	是[] 否[] 免[]	
3.9	较长的语句 (>80 字符) 要分成多行书写, 长表达式要在低优先级操作符处划分新行, 操作符放在新行之首, 划分出的新行要进行适当的缩进, 使排版整齐, 语句可读	是[] 否[] 免[]	
四: 程序的注释			
4.1	注释行数不得低于代码行数的 60%	是[] 否[] 免[]	
4.2	尽量把注释放在代码行的最后, 如果一行当中代码过长影响注释查看的, 把注释放在代码行的上一行, 不可放在下方	是[] 否[] 免[]	
4.3	边写代码边注释, 修改代码同时修改相应的注释, 以保证注释与代码的一致性。不再有有用的注释要删除	是[] 否[] 免[]	
4.4	尽量避免在注释中使用缩写, 特别是不常用缩写	是[] 否[] 免[]	
4.5	当代码比较长, 特别是有多重嵌套时, 应当在一些段落的结束处加注释, 便于阅读	是[] 否[] 免[]	
4.6	文件 (如头文件.h 文件、.inc 文件、源文件等) 头部应进行注释, 注释必须列出: 版权说明、版本号、生成日期、作者、内容、功能、与其它文件的关系、修改日志等, 头文件的注释中还应包含函数功能简要说明; 版本改变时保留上一版本的文件注释。	是[] 否[] 免[]	
4.7	函数头部应进行注释, 列出: 函数的目的/功能、输入参数、输出参数、返回值、等。	是[] 否[] 免[]	
五: 命名规则			
5.1	程序中不要出现标识符完全相同的局部变量和全局变量, 尽管两者的作用域不同而不会发生语法错误, 但会使人误解	是[] 否[] 免[]	
5.2	尽量避免名字中出现数字编号, 如 Value1, Value2 等, 除非逻辑上的确需要编号	是[] 否[] 免[]	
5.3	使用英文缩写命名, 避免名称的拼音写法	是[] 否[] 免[]	
5.4	宏定义命名规则: M_ (大写的定义名称)	是[] 否[] 免[]	
5.5	typedef 定义命名规则: TP_ (大写的定义名称)	是[] 否[] 免[]	
5.6	常量命名规则: C_ (大写的定义名称)	是[] 否[] 免[]	
5.7	函数名称命名规则: 全部小写	是[] 否[] 免[]	
5.8	变量定义命名规则: (b, c, i, l, f, s, u)_ (小写的定义名称)	是[] 否[] 免[]	
六: 表达式和基本语句			
6.1	如果代码行中的运算符比较多, 用括号确定表达式的操作顺序, 避免使用默认的优先级	是[] 否[] 免[]	
6.2	在多重循环中, 如果有可能, 应当将最长的循环放在最内层, 最短的循环放在最外层, 以减少 CPU 跨切循环层的次数	是[] 否[] 免[]	
6.3	不可在 for 循环体内修改循环变量, 防止 for 循环失去控制	是[] 否[] 免[]	
6.4	不要忘记最后那个 default 分支。即使程序真的不需要 default 处理, 也应该保留语句 default : break	是[] 否[] 免[]	
6.5	需要对外公开的常量放在头文件中, 不需要对外公开的常量放在定义文件的头部。为便于管理, 可以把不同模块的常量集中存放在一个公共的头文件中	是[] 否[] 免[]	
6.6	函数内超过两处相同的常量必须使用宏定义	是[] 否[] 免[]	
6.7	如果某一常量与其它常量密切相关, 应在定义中包含这种关系, 而不应给出一些孤立的值	是[] 否[] 免[]	
6.8	宏定义必须加括号	是[] 否[] 免[]	
6.9	用括号明确表达式的操作顺序, 避免使用默认优先级	是[] 否[] 免[]	
6.10	存在隐式转换的语句必须强制转换	是[] 否[] 免[]	
七: 函数设计			
7.1	参数的书写要完整, 不要贪图省事只写参数的类型而省略参数名字。如果函数没有参数, 则用 void 填充	是[] 否[] 免[]	
7.2	必要时在函数体的“入口处”, 对参数的有效性进行检查	是[] 否[] 免[]	
7.3	必要时在函数体的“出口处”, 对 return 语句的正确性和效率进行检查	是[] 否[] 免[]	
7.4	避免函数有太多的参数, 参数个数尽量控制在 5 个以内	是[] 否[] 免[]	
7.5	函数的功能要单一, 不要设计多用途的函数	是[] 否[] 免[]	

序号	总 则 条 款	执行情况	说明
7.6	养成“使用调试器逐步跟踪程序”的习惯，只有这样才能发现问题的本质	是[] 否[] 免[]	
7.7	函数完成后必须检查所用使用变量的同步问题	是[] 否[] 免[]	
八：软件使用规范			
8.1	编译前应检查自己的Keil版本，保证版本应和技术部指定版本保持一致	是[] 否[] 免[]	
8.2	统一使用 UltraEdit 软件进行代码编辑	是[] 否[] 免[]	

[illegible]