**EDS事业部嵌入式软件**

**C语言编程规范**

**单位：**融硅思创(北京)科技有限公司

**部门：** 嵌入式软件部

**员工：**

**时间：** 年 月 日

修订历史

|  |  |  |
| --- | --- | --- |
| **版本** | **日期** | **原因** |
| V1.0 | 2017.6.12 | 创建 |
|  |  |  |

目 录

[1 规范制定说明 8](#_Toc486083217)

[1.1 前言 8](#_Toc486083218)

[1.2 原则 8](#_Toc486083219)

[1.3 术语 8](#_Toc486083220)

[2 工程名称 9](#_Toc486083221)

[2.1 【建议】工程名称的命名应该简洁明了，尽量保持在10个字符以内。 9](#_Toc486083222)

[2.2 【必须】该工程产生的所有相关文件(如\*.out、\*.bin)的名称均需要和工程名称一致。 9](#_Toc486083223)

[3 工程文件 10](#_Toc486083224)

[3.1 【必须】 工程文件需要至少划分为4个文件夹：App、Bsp、Driver、Output。 10](#_Toc486083225)

[4 版本发布 11](#_Toc486083226)

[4.1 【原则】软件版本命名要遵循《版本命名规则.pdf》。 11](#_Toc486083227)

[4.2 【原则】版本提交测试前要经过严格自测。 11](#_Toc486083228)

[4.3 【必须】要详细记录软件变更内容，相对于前一个版本具体的变更细节。 11](#_Toc486083229)

[4.4 【必须】软件版本源码的变更内容都要有对应的问题单号，可查询。 11](#_Toc486083230)

[4.5 【必须】软件版本正式发布时，版本发布包里至少要包含bootloader bin、app bin、界面、升级指导及发布变更记录。 11](#_Toc486083231)

[4.6 【原则】软件版本正式发布时，遵循标准化的版本发布流程。 11](#_Toc486083232)

[5 头文件 11](#_Toc486083233)

[5.1 【原则】头文件中适合放置接口的声明，不适合放置实现。 11](#_Toc486083234)

[5.2 【原则】头文件应当职责单一。 11](#_Toc486083235)

[5.3 【必须】每一个.c文件应有一个同名.h文件，用于声明需要对外公开的接口。 11](#_Toc486083236)

[5.4 【必须】禁止头文件循环依赖。 11](#_Toc486083237)

[5.5 【必须】禁止在头文件中定义变量。 11](#_Toc486083238)

[5.6 【必须】只能通过包含头文件的方式使用其他.c提供的接口，禁止在.c中通过extern的方式使用外部函数接口、变量。 11](#_Toc486083239)

[5.7 【必须】总是编写内部#include保护符（#define保护）。 11](#_Toc486083240)

[6 函数 12](#_Toc486083241)

[6.1 【原则】一个函数仅完成一件功能。 12](#_Toc486083242)

[6.2 【原则】重复代码应该尽可能提炼成函数。 12](#_Toc486083243)

[6.3 【必须】避免函数过长，新增函数不超过50行（非空非注释行）。 12](#_Toc486083244)

[6.4 【必须】可重入函数应避免使用共享变量；若需要使用，则应通过互斥手段（关中断、信号量）对其加以保护。 12](#_Toc486083245)

[6.5 【必须】函数调用过程中，需要由函数的调用者对函数参数的合法性负责。即：调用函数时，函数内部一般情况下不再对输入参数进行合法性检查。需要函数的调用者来保证函数的输入参数是合法的。 12](#_Toc486083246)

[6.6 【必须】对调用函数的返回错误代码要仔细、全面的处理。不要因为程序中不可能出现该错误就不去处理。 12](#_Toc486083247)

[6.7 【必须】废弃代码（没有被调用的函数和变量)要及时清除。 13](#_Toc486083248)

[6.8 【建议】函数不变参数使用const。 13](#_Toc486083249)

[6.9 【建议】if语句、else if语句，都需要在后面加上else语句。 13](#_Toc486083250)

[6.10 【必须】switch语句中，需要有default。 13](#_Toc486083251)

[6.11 【建议】尽量使用乘法或其他方法来代替除法。尽量避免浮点运算的除法。 14](#_Toc486083252)

[6.12 【必须】一个运算过程中如果有乘法有除法，应先算乘法，最后计算除法。 14](#_Toc486083253)

[6.13 【必须】除法运算要避免出现除以0的错误。在做除法运算之前，最好对除数做检查。 14](#_Toc486083254)

[6.14 【必须】在使用内存时要注意，不要让调用的内存越界。 14](#_Toc486083255)

[6.15 【建议】函数的参数个数不超过5个。 14](#_Toc486083256)

[7 变量 14](#_Toc486083257)

[7.1 【原则】一个变量只有一个功能，不能把一个变量用作多种用途。 14](#_Toc486083258)

[7.2 【必须】防止局部变量和全局变量同名。 14](#_Toc486083259)

[7.3 【建议】可以在全局变量的命名时，名字前面加入“g\_”来代表全局变量。 14](#_Toc486083260)

[7.4 【建议】一般情况下，全局变量在声明的同时，就需要对全局变量赋值。 15](#_Toc486083261)

[7.5 【必须】未赋值的变量不可以进行运算。 15](#_Toc486083262)

[7.6 【建议】设计数据结构时，要注意元素的布局与排列顺序。尽量减少数据结构占用的空间。 15](#_Toc486083263)

[7.7 【原则】结构功能单一，不要设计面面俱到的数据结构。 15](#_Toc486083264)

[7.8 【必须】宏的定义需要全部使用大写字母。 15](#_Toc486083265)

[7.9 【建议】不用或少用全局变量。使用过多的全局变量会影响到程序的移植。 15](#_Toc486083266)

[7.10 【必须】单个文件内部使用的全局变量要在声明时加入static。 15](#_Toc486083267)

[7.11 【必须】跨文件使用的全局变量和函数，要在变量或函数声明的.c文件对应的.h文件中，加入extern 函数或变量名。 15](#_Toc486083268)

[7.12 【建议】尽量减少没有必要的数据类型默认转换与强制转换。 15](#_Toc486083269)

[8 宏、常量 15](#_Toc486083270)

[8.1 【必须】用宏定义表达式时，要使用完备的括号。 15](#_Toc486083271)

[8.2 【必须】将宏所定义的多条表达式放在大括号中。 15](#_Toc486083272)

[8.3 【必须】使用宏时，不允许参数发生变化。 16](#_Toc486083273)

[8.4 【必须】在程序中不允许直接使用魔鬼数字。应尽量使用有意义的枚举或宏定义来标明。 16](#_Toc486083274)

[8.5 【建议】宏定义中尽量不使用return、goto、continue、break等改变程序流程的语句。 16](#_Toc486083275)

[8.6 【建议】常量建议使用const定义代替宏。 16](#_Toc486083276)

[8.7 【建议】如果某个常量在多个函数中使用（超过3个函数），则必须将该常量定义为宏或枚举。 16](#_Toc486083277)

[8.8 【建议】和硬件外设相关的常量（如地址、命令等）需要定义为宏或枚举。 16](#_Toc486083278)

[8.9 【建议】在整个程序代码中，尽量避免出现硬件接口的描述。如需要使用硬件接口，则通过宏或枚举来实现。 16](#_Toc486083279)

[8.10 【建议】硬件的IO接口尽量全部封装成宏定义，以便于修改。 16](#_Toc486083280)

[9 表达式 16](#_Toc486083281)

[9.1 【建议】 函数调用不要作为另一个函数的参数使用，否则对于代码的调试、阅读都不利。 16](#_Toc486083282)

[9.2 【建议】赋值语句不要写在if等语句中。 17](#_Toc486083283)

[9.3 【建议】赋值语句不要作为函数的参数使用。 17](#_Toc486083284)

[9.4 【建议】用括号明确表达式的操作顺序，避免过分依赖默认优先级。 17](#_Toc486083285)

[9.5 【建议】赋值操作符不能使用在产生布尔值的表达式上。 18](#_Toc486083286)

[10 排版 18](#_Toc486083287)

[10.1 【必须】程序采用缩进风格编写，推荐缩进的字符数为4个空格。 18](#_Toc486083288)

[10.2 【建议】相对独立的程序块之间、变量说明之后需要加空行。 18](#_Toc486083289)

[10.3 【必须】较长的语句（大于80个字符），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。划分出的新行要进行适当的缩进。使排版整齐，程序结构清晰，可读性强。 18](#_Toc486083290)

[10.4 【必须】不允许将多个语句放到同一行。一行只能写一条语句。 19](#_Toc486083291)

[10.5 【必须】if、for、do、while、case、switch、default语句独占一行，且if、for、do、while语句的执行语句部分无论多少都要加大括号{}。 19](#_Toc486083292)

[10.6 【必须】对齐只使用空格键，不使用TAB键。 19](#_Toc486083293)

[10.7 【必须】程序块的分界符（如大括号{}）应各自独占一行且位于同一列。并与引用他们的语句左对齐。分界符内部的代码应与分界符保持一个缩进的距离。 19](#_Toc486083294)

[10.8 【必须】如果代码行中运算符较多，需要用括号确定表达式的操作顺序。避免使用默认的优先级。 20](#_Toc486083295)

[11 注释 20](#_Toc486083296)

[11.1 【原则】注释的内容要清楚、明了，含义准确，防止注释二义性。 20](#_Toc486083297)

[11.2 【原则】在代码的功能、意图层次上进行注释，即注释解释代码难以直接表达的意图，而不是重复描述代码。 20](#_Toc486083298)

[11.3 【原则】修改代码时，维护代码周边的所有注释，以保证注释与代码的一致性。不再有用的注释要删除。 20](#_Toc486083299)

[11.4 【原则】避免在注释中使用缩写，除非是业界通用或子系统内标准化的缩写。 20](#_Toc486083300)

[11.5 【原则】同一产品或项目组统一注释风格。 20](#_Toc486083301)

[11.6 【必须】一般情况下，注释应占程序代码的20%以上。 20](#_Toc486083302)

[11.7 【必须】文件的头需要进行注释，主要描述该文件的名称、功能、版本号、修改日志等信息。文件头的注释需要放到该文件的最上面。 20](#_Toc486083303)

[11.8 【必须】函数的头部需要进行注释，主要描述该函数的名称、功能、输入参数、返回参数以及备注信息。对于输入参数，需要明确说明输入参数的取值范围，用于给函数调用者以参考。函数头的注释需要放到对应函数的上面。 21](#_Toc486083304)

[11.9 【建议】对于定义的变量、常量，在声明时最好加入注释说明该变量、常量的功能。 21](#_Toc486083305)

[11.10 【必须】对于全局变量，则必须注释说明该全局变量的功能、取值范围以及以及存取时注意事项等的说明。 21](#_Toc486083306)

[11.11 【必须】对代码的注释应该距离相应代码很近，注释应放在其代码上方相邻位置或右方，不可放在下面。如放于上方则需与其上面的代码用空行隔开，且与下方代码缩进相同。 21](#_Toc486083307)

[11.12 【必须】对数据结构的声明，需要在数据结构声明的上方相邻位置注释该数据结构的功能。 21](#_Toc486083308)

[11.13 【建议】数据结构中的每个域如果其命名无法充分描述其功能的，则需要在对应域的右方注释说明该域的功能。 21](#_Toc486083309)

[11.14 【必须】避免在一行代码或表达式的中间插入注释。 21](#_Toc486083310)

[11.15 【建议】如代码段较长（一页显示不下），或存在多重嵌套时（三重以上），需要在程序块的结束行右侧加以注释标记。如下所示： 21](#_Toc486083311)

[11.16 【建议】代码的注释尽量使用中文。 22](#_Toc486083312)

[12 可读性 22](#_Toc486083313)

[12.1 【建议】在程序中尽量避免使用不可理解的数字，应尽量使用有意义的枚举或宏定义来标明。 22](#_Toc486083314)

[12.2 【建议】程序中关系较为紧密的代码应相邻，各个功能模块之间要用空行分隔开来。 22](#_Toc486083315)

[13 可移植性 23](#_Toc486083316)

[13.1 【必须】不能定义、重定义或取消定义标准库/平台中保留的标识符、宏和函数。 23](#_Toc486083317)

[13.2 【建议】不使用与硬件或操作系统关系很大的语句，而使用建议的标准语句，以提高软件的可移植性和可重用性。 23](#_Toc486083318)

[13.3 【建议】除非为了满足特殊需求，避免使用嵌入式汇编。 23](#_Toc486083319)

[14 代码编辑、编译 23](#_Toc486083320)

[14.1 【必须】使用编译器的最高告警级别，理解所有的告警，通过修改代码而不是降低告警级别来消除所有告警。 23](#_Toc486083321)

[14.2 【必须】在产品软件（项目组）中，要统一编译开关、静态检查选项以及相应告警清除策略。 23](#_Toc486083322)

[15 可测试性 23](#_Toc486083323)

[15.1 【原则】模块划分清晰，接口明确，耦合性小，有明确输入和输出，否则单元测试实施困难。 23](#_Toc486083324)

[15.2 【必须】统一的系统调测开关及相应打印函数，并且要有详细的说明。 23](#_Toc486083325)

[15.3 【必须】统一的日志格式及日志规范，重点包括日志分类、级别、关键记录内容。代码关键分支要记录日志。 23](#_Toc486083326)

# 规范制定说明

## 前言

为提高产品代码质量及规范性，为软件开发人员的代码编写提供参考依据和统一标准。特建立此适合EDS产品的软件编码规范，规范涉及编码风格、语法说明、注意事项等等，该编码规范EDS事业部所有软件人员共同参与制定。

## 原则

代码编写应遵从如下原则：

### 简洁

简洁就是易于理解并且易于实现。

* 代码越长越多，越容易引入错误，也意味着可靠性越低。
* 废弃的代码要及时清除。
* 重复代码应该尽可能提炼成函数。

### 清晰

清晰就是要做到易于维护，易于重构。

### 风格

风格统一远比“为了统一风格”要好。

修改或重构其他风格代码时，最好的做法是继续保持原有代码风格。

## 术语

原则：编程时必须坚持的指导思想。

必须：编程时强制必须遵守的约定。

建议：编程时必须加以考虑的约定。

说明：对内容进行必要的解释。

示例：对此原则/必须/建议从正、反两个方面给出例子。

# 工程名称

## 【建议】工程名称的命名应该简洁明了，尽量保持在10个字符以内。

举例：

起爆电压测试仪工程命名：VolTest

新300型起爆器工程命名：New300

## 【必须】该工程产生的所有相关文件(如\*.out、\*.bin)的名称均需要和工程名称一致。

举例：

新300型起爆器的工程名称为：New300.eww，则工程产生的相关文件也应为

New300.out、New300.bin。

# 工程文件

## 【必须】 工程文件需要至少划分为4个文件夹：App、Bsp、Driver、Output。

### 文件夹App

用于存放应用层的程序，基本上不直接和底层硬件打交道，主要用于搭建流程、事件、状态等。

main函数需要放到该文件夹中的.c文件中。如果程序中使用了uCOS-II，则需要将uCOS-II的所有进程都放到App文件夹中的.c文件中。

App文件夹中的所有文件均需要以App开头来命名文件。如AppLcd.c、AppInit.c等文件。

### 文件夹Bsp

用于存放MCU相关的程序，以及启动程序(\*.s)。如startup\_stm32f2xx.s、stm32f2xx\_adc.c等文件。

### 文件夹Driver

用于存放驱动层的程序。该部分程序是直接和底层硬件打交道的代码。将底层硬件的功能封装成一个个的函数，供应用层调用。

Driver文件夹中的所有文件均需要以Drv开头来命名文件。如DrvLcd.c、DrvFlash.c等文件。

### 文件夹Output

用于存放工程的输出文件，如\*.map、\*.out。文件。

### 其他文件夹

文件夹名称为Third。在该部分添加第三方代码。。如uCOS-II内核，则添加uCOS-II文件夹，里面包含和uCOS-II相关的\*.c文件。如FAT文件系统，则添加FATFS文件夹，包含和FATFS相关的\*.c文件。

# 版本发布

## 【原则】软件版本命名要遵循《版本命名规则.pdf》。

## 【原则】版本提交测试前要经过严格自测。

## 【必须】要详细记录软件变更内容，相对于前一个版本具体的变更细节。

## 【必须】软件版本源码的变更内容都要有对应的问题单号，可查询。

## 【必须】软件版本正式发布时，版本发布包里至少要包含bootloader bin、app bin、界面、升级指导及发布变更记录。

## 【原则】软件版本正式发布时，遵循标准化的版本发布流程。

# 头文件

## 【原则】头文件中适合放置接口的声明，不适合放置实现。

## 【原则】头文件应当职责单一。

## 【必须】每一个.c文件应有一个同名.h文件，用于声明需要对外公开的接口。

## 【必须】禁止头文件循环依赖。

说明：头文件循环依赖，指a.h包含b.h，b.h包含c.h，c.h包含a.h之类导致任何一个头文件修改，都导致所有包含了a.h/b.h/c.h的代码全部重新编译一遍。而如果是单向依赖，如a.h包含b.h，b.h包含c.h，而c.h不包含任何头文件，则修改a.h不会导致包含了b.h/c.h的源代码重新编译。

## 【必须】禁止在头文件中定义变量。

说明：在头文件中定义变量，将会由于头文件被其他.c文件包含而导致变量重复定义。

## 【必须】只能通过包含头文件的方式使用其他.c提供的接口，禁止在.c中通过extern的方式使用外部函数接口、变量。

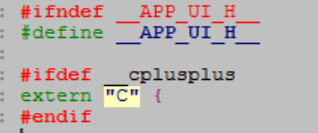
## 【必须】总是编写内部#include保护符（#define保护）。

说明：所有头文件都应当使用#define防止头文件被多重包含，命名格式为\_\_FILENAME\_H\_\_。

定义包含保护符时，应该遵守如下规则：

1）保护符使用唯一名称；

2）不要在受保护部分的前后放置代码或者注释。



# 函数

## 【原则】一个函数仅完成一件功能。

说明：一个函数实现多个功能给开发、使用、维护都带来很大的困难。

将没有关联或者关联很弱的语句放到同一函数中，会导致函数职责不明确，难以理解，难以测试和改动。

## 【原则】重复代码应该尽可能提炼成函数。

说明：重复代码提炼成函数可以带来维护成本的降低。

## 【必须】避免函数过长，新增函数不超过50行（非空非注释行）。

说明：本规则仅对新增函数做要求，对已有函数修改时，建议不增加代码行。

过长的函数往往意味着函数功能不单一，过于复杂，这样将会给代码阅读及维护带来不便。

## 【必须】可重入函数应避免使用共享变量；若需要使用，则应通过互斥手段（关中断、信号量）对其加以保护。

说明：可重入函数是指可能被多个任务并发调用的函数。在多任务操作系统中，函数具有可重入性是多个任务可以共用此函数的必要条件。共享变量指的全局变量和static变量。

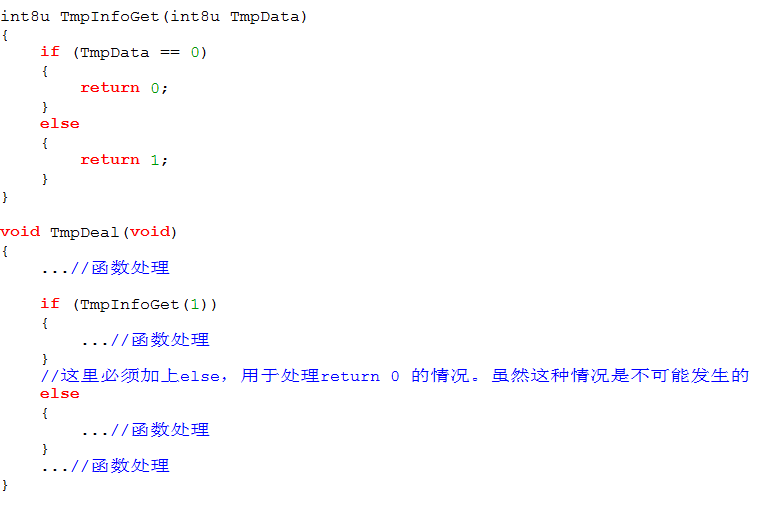
编写C语言的可重入函数时，不应使用static局部变量，否则必须经过特殊处理，才能使函数具有可重入性。

## 【必须】函数调用过程中，需要由函数的调用者对函数参数的合法性负责。即：调用函数时，函数内部一般情况下不再对输入参数进行合法性检查。需要函数的调用者来保证函数的输入参数是合法的。

## 【必须】对调用函数的返回错误代码要仔细、全面的处理。不要因为程序中不可能出现该错误就不去处理。

说明：这么做可以保证程序的严谨性，并且有利于以后程序的修改。

举例：



## 【必须】废弃代码（没有被调用的函数和变量)要及时清除。

## 【建议】函数不变参数使用const。

## 【建议】if语句、else if语句，都需要在后面加上else语句。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
| C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\%)[O}58_OHZOAEC(K]BLKIS.png | C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\0WA6BL5IIJX74FHQIUN7T%F.png |

## 【必须】switch语句中，需要有default。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
| C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\D%}`4JT4BXI1I1AMM3J)BEX.png | C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\4QEOKS)2HJA9JG_`L77HC54.png |

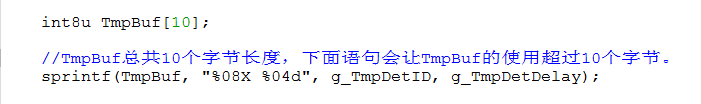
## 【建议】尽量使用乘法或其他方法来代替除法。尽量避免浮点运算的除法。

## 【必须】一个运算过程中如果有乘法有除法，应先算乘法，最后计算除法。

## 【必须】除法运算要避免出现除以0的错误。在做除法运算之前，最好对除数做检查。

## 【必须】在使用内存时要注意，不要让调用的内存越界。

举例：



## 【建议】函数的参数个数不超过5个。

说明：函数的参数过多，会使得该函数易于受外部（其他部分的代码）变化的影响，从而影响维护工作。函数的参数过多同时也会增大测试的工作量。函数的参数个数不要超过5个，如果超过了建议拆分为不同函数。

# 变量

## 【原则】一个变量只有一个功能，不能把一个变量用作多种用途。

说明：一个变量只用来表示一个特定功能，不能把一个变量作多种用途，即同一变量取值不同时，其代表的意义也不同。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
| Locate具有两种功能：位置和函数DeleteFromQue的返回值 |  |

## 【必须】防止局部变量和全局变量同名。

## 【建议】可以在全局变量的命名时，名字前面加入“g\_”来代表全局变量。

举例：

unsigned char g\_DetInfoState = 0;

## 【建议】一般情况下，全局变量在声明的同时，就需要对全局变量赋值。

## 【必须】未赋值的变量不可以进行运算。

## 【建议】设计数据结构时，要注意元素的布局与排列顺序。尽量减少数据结构占用的空间。

## 【原则】结构功能单一，不要设计面面俱到的数据结构。

## 【必须】宏的定义需要全部使用大写字母。

## 【建议】不用或少用全局变量。使用过多的全局变量会影响到程序的移植。

## 【必须】单个文件内部使用的全局变量要在声明时加入static。

## 【必须】跨文件使用的全局变量和函数，要在变量或函数声明的.c文件对应的.h文件中，加入extern 函数或变量名。

举例：

如果在Flash.c文件中定义：

全局变量：int8u g\_State = 0;

函数：void FlashWrite(void);

如果该全局变量和函数可以被别的文件调用，则需要在Flash.h文件中加入如下代码：

extern int8u g\_State;

extern void FlashWrite(void);

如果在Ui.c文件中需要使用该全局变量和该函数，则只需要在Ui.c文件的开始位置加入如下代码即可：

#include “Flash.h”

## 【建议】尽量减少没有必要的数据类型默认转换与强制转换。

# 宏、常量

## 【必须】用宏定义表达式时，要使用完备的括号。

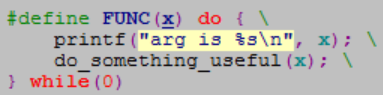
说明：因为宏只是简单的代码替换，不会像函数一样先将参数计算后，再传递。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
|  |  |

## 【必须】将宏所定义的多条表达式放在大括号中。

说明：更好的方法是多条语句写成do while(0)的方式。

举例：



## 【必须】使用宏时，不允许参数发生变化。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
| #define A(a) ((a) \* (a))  int a = 5;  int b;  b = SQUARE(a++);  // 结果：a = 7，即执行了两次增。 | #define A(a) ((a) \* (a))  int a = 5;  int b;  b = SQUARE(a);  a++;  // 结果： a = 6，即只执行了一次增。 |

## 【必须】在程序中不允许直接使用魔鬼数字。应尽量使用有意义的枚举或宏定义来标明。

## 【建议】宏定义中尽量不使用return、goto、continue、break等改变程序流程的语句。

## 【建议】常量建议使用const定义代替宏。

## 【建议】如果某个常量在多个函数中使用（超过3个函数），则必须将该常量定义为宏或枚举。

说明：这么做的目的是为了函数比较好修改。

## 【建议】和硬件外设相关的常量（如地址、命令等）需要定义为宏或枚举。

## 【建议】在整个程序代码中，尽量避免出现硬件接口的描述。如需要使用硬件接口，则通过宏或枚举来实现。

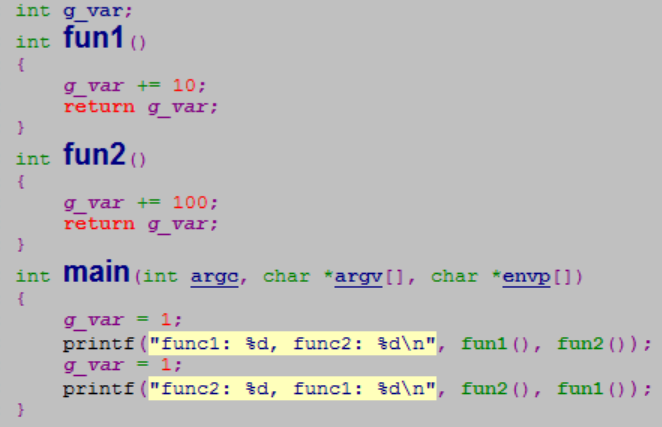
C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\XRAZE@B8NF26`8Q3(I%K)0Q.png

## 【建议】硬件的IO接口尽量全部封装成宏定义，以便于修改。

# 表达式

## 【建议】 函数调用不要作为另一个函数的参数使用，否则对于代码的调试、阅读都不利。

说明：如下代码不合理，仅用于说明当函数作为参数时，由于参数压栈次数不是代码可以控制的，可能造成未知的输出：

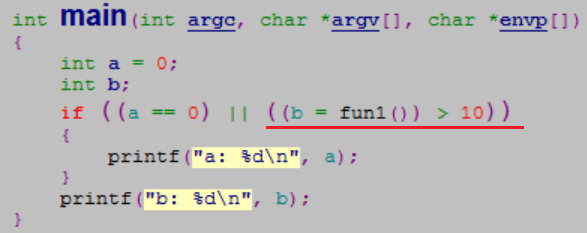


上面的代码，使用断点调试起来也比较麻烦，可读性也差，所以不要为了节约代码行而写这种代码，反而找出适得其反。

## 【建议】赋值语句不要写在if等语句中。

说明：因为if语句中，会根据条件依次判断，如果前一个条件已经可以判定整个条件，则后续条件语句不会再运行，所以可能导致期望的部分赋值没有得到运行。

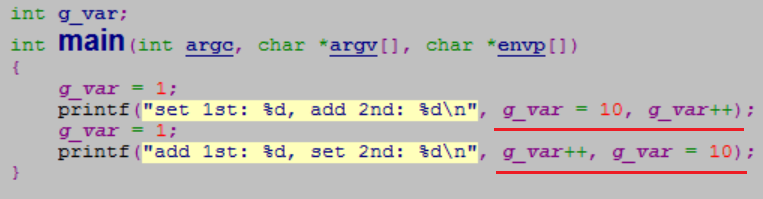
示例：



## 【建议】赋值语句不要作为函数的参数使用。

说明：赋值语句作为函数参数来使用，参数的压栈顺序不同可能导致结果未知。

示例：如下代码可读性很差。



## 【建议】用括号明确表达式的操作顺序，避免过分依赖默认优先级。

说明：使用括号强调所使用的操作符，防止因默认的优先级与设计思想不符而导致程序出错，同时使得代码更为清晰可读，然而过多的括号会分散代码使其降低了可读性。

1.一元操作符，不需要使用括号

x = ~a; /\* 一元操作符，不需要括号\*/

x = -a; /\* 一元操作符，不需要括号\*/

2.二元以上操作符，如果涉及多种操作符，则应该使用括号

x = a + b + c; /\* 操作符相同，不需要括号\*/

x = f ( a + b, c ) /\* 操作符相同，不需要括号\*/

if (a && b && c) /\* 操作符相同，不需要括号\*/

x = (a \* 3) + c + d; /\* 操作符不同，需要括号\*/

x = ( a == b ) ? a : ( a –b ); /\* 操作符不同，需要括号\*/

## 【建议】赋值操作符不能使用在产生布尔值的表达式上。

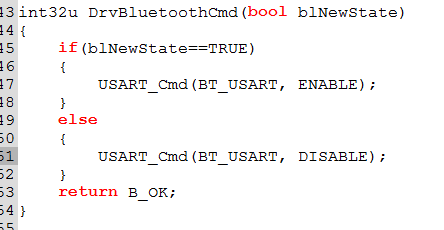
说明：如果布尔值表达式需要赋值操作，那么赋值操作必须在操作数之外分别进行。 这可以帮助避免=和==的混淆，帮助我们静态地检查错误。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
| If((x=y)!=0)  {  foo();  } | x=y;  if(x!=0)  {  foo ();  } |

# 排版

## 【必须】程序采用缩进风格编写，推荐缩进的字符数为4个空格。

举例：



## 【建议】相对独立的程序块之间、变量说明之后需要加空行。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
|  |  |

## 【必须】较长的语句（大于80个字符），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。划分出的新行要进行适当的缩进。使排版整齐，程序结构清晰，可读性强。

C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\B`([0_0E7TSH`1Y2}08_)[E.png

在低优先级操作符处划分新行

新行进行适当的缩进

## 【必须】不允许将多个语句放到同一行。一行只能写一条语句。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
|  |  |

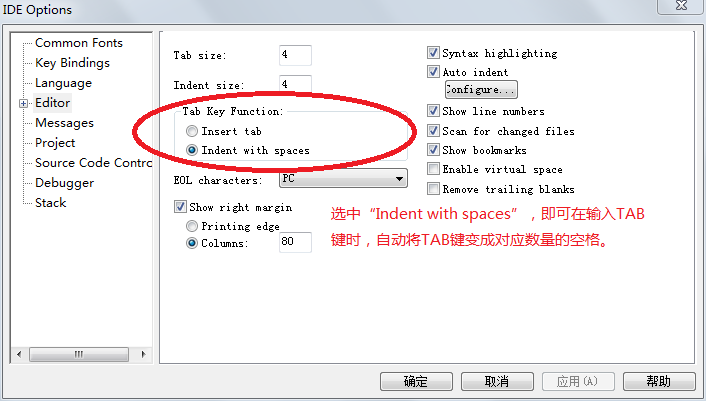
## 【必须】if、for、do、while、case、switch、default语句独占一行，且if、for、do、while语句的执行语句部分无论多少都要加大括号{}。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
|  |  |

## 【必须】对齐只使用空格键，不使用TAB键。

原因：由于TAB键所设置的空格数目不同会造成排版的混乱。故规定不许使用TAB键。程序的对齐和缩进只使用空格。

可以通过设置修改按下TAB键，插入空格。如下所示（IAR）：



## 【必须】程序块的分界符（如大括号{}）应各自独占一行且位于同一列。并与引用他们的语句左对齐。分界符内部的代码应与分界符保持一个缩进的距离。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
|  |  |

## 【必须】如果代码行中运算符较多，需要用括号确定表达式的操作顺序。避免使用默认的优先级。

原因：这样做会提高程序的可读性，并且可以避免出现因为运算优先级的顺序出错而导致的程序出错。

# 注释

## 【原则】注释的内容要清楚、明了，含义准确，防止注释二义性。

## 【原则】在代码的功能、意图层次上进行注释，即注释解释代码难以直接表达的意图，而不是重复描述代码。

## 【原则】修改代码时，维护代码周边的所有注释，以保证注释与代码的一致性。不再有用的注释要删除。

说明：不要将无用的代码留在注释中，随时可以从源代码配置库中找回代码；即使只是想暂时排除代码，也要留个标注，不然可能会忘记处理它。

## 【原则】避免在注释中使用缩写，除非是业界通用或子系统内标准化的缩写。

## 【原则】同一产品或项目组统一注释风格。

## 【必须】一般情况下，注释应占程序代码的20%以上。

## 【必须】文件的头需要进行注释，主要描述该文件的名称、功能、版本号、修改日志等信息。文件头的注释需要放到该文件的最上面。

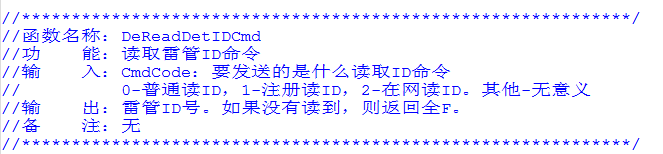
下面是一个文件头的注释，描述的内容不要少于下面所列。



说明：文件头的注释，主要用在驱动层文件中。由于驱动层文件对程序整体比较重要，修改相对来说较少。故必须存在文件头，来对文件整体有一个详细的说明。在应用层文件中，由于修改会较频繁，可以不需要文件头的注释。

## 【必须】函数的头部需要进行注释，主要描述该函数的名称、功能、输入参数、返回参数以及备注信息。对于输入参数，需要明确说明输入参数的取值范围，用于给函数调用者以参考。函数头的注释需要放到对应函数的上面。

下面是一个函数头的注释，描述的内容不要少于下面所列。



## 【建议】对于定义的变量、常量，在声明时最好加入注释说明该变量、常量的功能。

## 【必须】对于全局变量，则必须注释说明该全局变量的功能、取值范围以及以及存取时注意事项等的说明。

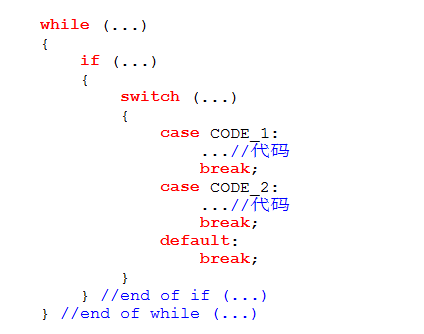
## 【必须】对代码的注释应该距离相应代码很近，注释应放在其代码上方相邻位置或右方，不可放在下面。如放于上方则需与其上面的代码用空行隔开，且与下方代码缩进相同。

## 【必须】对数据结构的声明，需要在数据结构声明的上方相邻位置注释该数据结构的功能。

## 【建议】数据结构中的每个域如果其命名无法充分描述其功能的，则需要在对应域的右方注释说明该域的功能。

## 【必须】避免在一行代码或表达式的中间插入注释。

## 【建议】如代码段较长（一页显示不下），或存在多重嵌套时（三重以上），需要在程序块的结束行右侧加以注释标记。如下所示：



原因：这样做会使代码更清晰，更便于阅读及修改。

## 【建议】代码的注释尽量使用中文。

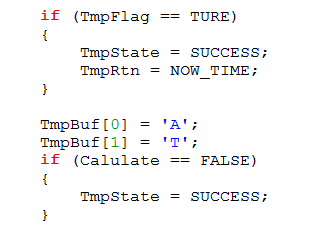
# 可读性

## 【建议】在程序中尽量避免使用不可理解的数字，应尽量使用有意义的枚举或宏定义来标明。

|  |  |
| --- | --- |
| 错误的书写方法： | 正确的书写方法： |
| C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\2}K4{OB`}4`R6VJZT6@E%D1.png | C:\Users\zhl0617\AppData\Roaming\Tencent\Users\120798488\QQ\WinTemp\RichOle\77WR5P9V`IQ23N{W5)Y4$BF.png |

## 【建议】程序中关系较为紧密的代码应相邻，各个功能模块之间要用空行分隔开来。

举例：



# 可移植性

## 【必须】不能定义、重定义或取消定义标准库/平台中保留的标识符、宏和函数。

## 【建议】不使用与硬件或操作系统关系很大的语句，而使用建议的标准语句，以提高软件的可移植性和可重用性。

说明：使用标准的数据类型，有利于程序的移植。

示例：如下例子（在DOS下BC3.1环境中），在移植时可能产生问题。

void main()

{

register int index; // 寄存器变量

\_AX = 0x4000; // \_AX是BC3.1提供的寄存器“伪变量”

... // program code

}

## 【建议】除非为了满足特殊需求，避免使用嵌入式汇编。

说明：程序中嵌入式汇编，一般都对可移植性有较大的影响。

# 代码编辑、编译

## 【必须】使用编译器的最高告警级别，理解所有的告警，通过修改代码而不是降低告警级别来消除所有告警。

## 【必须】在产品软件（项目组）中，要统一编译开关、静态检查选项以及相应告警清除策略。

说明：如果必须禁用某个告警，应尽可能单独局部禁用，并且编写一个清晰的注释，说明为什么屏蔽。某些语句经编译/静态检查产生告警，但如果你认为它是正确的，那么应通过某种手段去掉告警信息。

# 可测试性

## 【原则】模块划分清晰，接口明确，耦合性小，有明确输入和输出，否则单元测试实施困难。

## 【必须】统一的系统调测开关及相应打印函数，并且要有详细的说明。

## 【必须】统一的日志格式及日志规范，重点包括日志分类、级别、关键记录内容。代码关键分支要记录日志。