两年半短视频系统

代码编写规范

两年半短视频公司/团队

2201338115 张俊强

2022.11.20

目录

[1. 排版规范 2](#_Toc121391428)

[2. 注释规范 2](#_Toc121391429)

[3. 命名规范 2](#_Toc121391430)

# 排版规范

**括号规范**

小括号规则  
a) 不要把小括号和关键词（ if 、 while 等） 紧贴在一起， 要用空格隔开它们。  
b) 不要把小括号和函数名紧贴在一起。  
c) 除非必要， 不要在 Return 返回语句中使用小括号。 因为关键字不是函数， 如果小括号紧贴  
着函数名和关键字， 二者很容易被看成是一体的。

|  |
| --- |
| 大括号规则 |
| 将大括号放置在关键词下方的同列处， 例如： |
| if ($condition) | while ($condition) |
| { | { |
| ... | ... |
| } | } |

**缩进规范**

缩进规则  
使用一个“ Tab” 为每层次缩进。 例如：  
function func()  
 {  
 if (something bad)  
 {  
 if (another thing bad)  
 {  
 while (more input)  
 {

}  
}

}

}

**对齐规范**

|  |
| --- |
| 变量的申明和初始化都应对齐。 例如： |

int m\_iCount;

int i,j;

float m\_flncome,m\_fPay;

m\_iCount = 0;

i = 1;

m\_flncome = 0.3;

# 注释规范

**文件注释规范**

对于自己创建的代码文件（如函数、脚本），在文件开头，一般编写如下注释:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FileName :

Copyright (c) 2004-xxxx \*\*\*\*\*\*\*\*\*公司技术开发部

writer:

create Date:

Rewriter:

Rewrite Date:

lmpact:

Main Content (Function Name、parameters、 returns)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**模块注释规范**

模块开始建议使用以下形式书写模块注释:

//<summary>

///Module ID:<模块编号，可以引用系统设计中的模块编号>

///Depiction:<对此类的描述，可以引用系统设计中的描述>

///Author:作者中文名

///Create Date:<模块创建日期，格式: YYYY-MM-DD>

///</summary>

如果模块只进行部分少量代码的修改时，则每次修改须添加以下注释:

///Rewriter: Rewrite Date:<修改日期，格式:

YYYY-MM-DD>Start1:

/\*原代码内容\*/

///Endl:

将原代码内容注释掉，然后添加新代码使用以下注释:

///Added by: Add date:<添加日期，格式: YYYY一MM-DD> Start2:

新代码内容

///End2:

如果模块输入输出参数或功能结构有较大修改，则每次修改必须添加以下注释:

///<summary>

**语句注释规范**

1、头文件包含Includes

2、私有类型定义 Private typedef

3、私有定义Private define

4、私有宏定义 Private macro

5、私有变量 Private variables

6、私有函数原型Private function prototypes

7、私有函数Private functions

8、私有函数前注释

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : FSMC\_NOR\_Init

\* Description : Configures the FSMC and GPIOs to interface with the NOR memory.

\* This function must be called before any write/read operation

\* on the NOR.

\* Input : None

\* Output : None

\* Return : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

9、程序块采用缩进风格编写，缩进空格为4。

10、相对独立的程序块之间、变量说明之后必须加空行；

11、较长的字符（>80字符）要分成多行书写，长表达式要在低优先级操作符划分新行，操作符放在新行之首，新行要恰当缩进，保持排版整齐；

12、循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分，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;

13、若函数或过程中的参数较长，则要进行适当的划分。

14、不允许把多个短语句写在一行中，即一行只写一条语句。

15、if、for、do、while、case、switch、default等语句自占一行，且if、for、do、while等语句的执行语句部分无论多少都要加括号{}。

16、对齐只使用空格键，不使用TAB键；

17、函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进风格，case语句下的情况处理语句也要遵从语句缩进要求

18、程序块的分界符（如C/C++语言的大括号‘{’和‘}’）应各独占一行并且位于同一列，同时与引用它们的语句左对齐。在函数体的开始、类的定义、结构的定义、枚举的定义以及if、for、do、while、switch、case语句中的程序都要采用如上的缩进方式。

19、在两个以上的关键字、变量、常量进行对等操作时，它们之间的操作符之前、之后或者前后要加空格；进行非对等操作时，如果是关系密切的立即操作符（如－>），后不应加空格；

20、一行程序以小于80字符为宜，不要写得过长。

21、一般情况下，源程序有效注释量必须在20％以上。注释不宜太多、不宜太少，准确易懂简洁；

22、说明性文件（如头文件.h文件、.inc文件、.def文件、编译说明文件.cfg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有函数功能简要说明。

23、源文件头部应进行注释，列出：版权说明、版本号、生成日期、作者、模块目的/功能、主要函数及其功能、修改日志等。

24、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、调用

关系（函数、表）等。

25、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

26、注释的内容要清楚、明了，含义准确，防止注释二义性。

说明：错误的注释不但无益反而有害；

27、避免在注释中使用缩写，特别是非常用缩写；

28、注释应与其描述的代码相近，对代码的注释应放在其上方或右方（对单条语句的注释）相邻位置，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。

29、数据结构声明(包括数组、结构、类、枚举等)，如果其命名不是充分自注释的，必须

加以注释。对数据结构的注释应放在其上方相邻位置，不可放在下面；对结构中的每个域的注释放在此域的右方；

30、全局变量要有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取它以及

存取时注意事项等的说明。

31、注释与所描述内容进行同样的缩排

32、将注释与其上面的代码用空行隔开；

33、对变量的定义和分支语句（条件分支、循环语句等）必须编写注释；

34、对于switch语句下的case语句，如果因为特殊情况需要处理完一个case后进入下一

个case处理，必须在该case语句处理完、下一个case语句前加上明确的注释；

35、通过对函数或过程、变量、结构等正确的命名以及合理地组织代码的结构；

在代码的功能、意图层次上进行注释，提供有用、额外的信息。

36、在程序块的结束行右方加注释标记，以表明某程序块的结束。

说明：当代码段较长，特别是多重嵌套时，这样做可以使代码更清晰，更便于阅读。

37、注释格式尽量统一，建议使用“/\* …… \*/”

38、注释应考虑程序易读及外观排版的因素，使用的语言若是中、英兼有的，建议多使用中文，除非能用非常流利准确的英文表达；

# 命名规范

**变量命名规范**

变量（ Variable） 命名

程序文件(\*.cs)中的变量命名

所有变量如果无前缀必须首字母小写，前缀无特殊要求，但必须是易于识别公认的做法，如’m\_'‘\_’,切毋自创缩写 鉴于大多数名称都是通过连接若干单词构造的，必须使用大小写混合的格式以简化它们的阅读。每个单词的第一个字母都是大写. 即使对于可能仅出现在几个代码行中的生存期很短的变量，仍然使用有意义的名称。仅对于短循环索引使用单字母变量名，如i 或j。 在变量名中使用互补对，如 min/max、 begin/end和 open/close。

0不要使用原义数字或原义字符串，如For (i = 1; i <= 7;i++)。而是使用命名常数，如 For(i = 1;i <= NUM\_DAYS\_IN\_WEEK ; i++)以便于维护和理解。

(B)控件命名

控件命名= Web控件缩写前缀+[“\_”]+名称

如:DataGrid dg\_UserList或dgUserlist

**模块/函数命名规范。**

类(Class）命名

1. 名字应该能够标识事物的特性。
2. 名字尽量不使用缩写，除非它是众所周知的。
3. 名字可以有两个或三个单词组成，但通常不应多于三个。
4. 在名字中，所有单词第一个字母大写。例如 IsSuperUser，包含ID的，ID可以全部大写如CustomerID。
5. 使用名词或名词短语命名类。
6. 少用缩写。
7. 不要使用下划线字符(\_)。

例: public class FileStream

public class Button

public class String

接口（Interface）命名

和类命名规范相同，唯一区别是接口在名字前加上“I”前缀

例:

interface IDBCommand;

interface IButton;

方法(Method）命名

和类命名规范相同。

命名空间(NameSpace）命名

和类命名规范相同。

属性(Property)命名

与类名命名规范相同。