软件在产品开发中扮演越来越重要的地位。

从软件开发角度看，一个软件的开发往往需要多人合作，或以后的自己和当前的自己合作；

从软件维护角度看，一个软件的声明周期中，80%的精力在于维护，而维护往往又是有多人共同完成的；

使用统一的编码规范，便于团队成员共同开发和维护。

## 0. General Rules

* Readability counts. 清晰第一。
* Be consistent. 风格统一。
* Don't repeat yourself. 拒绝重复。
* Flat is better than nested. 逻辑扁平。
* Simple is better than complex. 简洁为美。

## 1. 命名规范

* 同一项目，格式统一，选择驼峰命名规则或小写+下划线命名规则；
* 变量名称不要过长，20字符以内

### 1.1 驼峰命名法则

函数，自定义类型采用大驼峰，变量采用小驼峰

bool TimerInit(void)  
{  
 uint8\_t index;  
}

### 1.2 小写+下划线命名法

* **函数和变量用小写+下划线命名**

bool timer\_init(void)  
{  
 uint8\_t index;  
}

* **类型定义**

使用typedef,定义类型以\_t结尾。

* **宏定义与常量**

大写字母与下划线组合

## 2. 编码格式

* 1 Tab == 4 spaces，用tab缩进；
* 双目操作符前后各留一个空格a + b，单目操作符不加空格!a；
* 大括号单独成行；
* if (a > b)  
  {  
   // other code here.  
  }
* if, while, for 后空一个写条件，如if (a > 0)
* 一行不超过80个字符，当长度较长时，分行写：
* void SendCanCmdRsp (int8u cmd,  
   int8u \*cmdDataPtr,  
   int8u \*cmdRspPtr);
* 一行只定义一个变量；
* 除头文件防止重复包含除外，避免使用 #ifdef or #ifndef. 用 defined() or !defined() 代替；
* 用int8\_t, uint32\_t等作为整型类型名，提高可移植性；
* 变量定义和逻辑代码间加空行；
* 不同逻辑块之间加空行；

## 3. 文件格式

### 3.1 头文件

头文件规范参考[h\_templete](templete.h)

* 向稳定方向包含；
* 禁止在头文件定义变量
* **头文件包含顺序**：先标准库头文件，后user 头文件，中间空一格
* #include <stdio.h>  
    
  #include "can.h"  
  #include "delay.h"
* 一个.c文件对应一个同名的.h文件

### 3.2 源文件

源文件规范参考[C\_templete](templete.c)

* 一个函数尽量仅完成一个功能；
* 重复代码提炼成函数；
* 避免函数过长，尽量小于50行；
* 避免嵌套过深，不超过4层，否则考虑把代码封装成函数；
* 废弃代码及时清除；
* 函数参数不变使用const限定。
* 函数的参数个数不超过5个。
* 尽量少用全局变量；
* 不允许使用魔鬼数字，可以定义成常量或宏

## 4. 注释

* 优秀的代码能够自我注释；
* 不写无用注释；
* 代码和注释需同步更新；
* 采用工具可以识别的注释格式：

### 4.1 使用doxgen自动生成漂亮的文档

#### 4.1.1 常用注释格式

* **不需要doxgen生成注释时**：

/\*  
 \* This is multi-line comments,  
 \* written in 2 lines (ok)  
 \*/

* **单行注释**

/\*! 单行注释 \*/  
uint16\_t position;

* **多行注释**

/\*!  
 \* 1、多行注释风格注释风格  
 \* 2、原结构体/联合体命名用下划线+小驼峰命名法，  
 \* 加下划线是为了避免不经意间和其它全局变量发生冲突，重定义后的名字用全大写  
 \* +下划线的方式，和宏定义的命名类似  
 \* 3、结构体/联合体定义时起始大括号放在行末，结束大括号放在行首，大括号和名称  
 \* 之间留一个空格  
 \*/

* **注释当前行**

uint16\_t position; /\*\*< 注释当前行 \*/

* **函数注释**

/\*!  
 \* \brief 主函数入口  
 \* 函数不同参数之间要有一个空格，且空格在逗号后面  
 \* 指针的\*星号靠近变量名，而不是靠近类型名；函数命名使用小写+下划线的  
 \* 方式，如get\_info()，不要使用驼峰命名法，如getInfo()或者GetInfo()  
 \*  
 \* \param [in] argc 运行可执行程序时后面带的参数个数（其中第0个参数默认  
 \* 是程序的全名），[in]代表的是输出参数，[out]代表是输出参数  
 \* [in out]代表既是输入又是输出参数  
 \* \param [in] argv 字符串指针数组，程序执行传入的参数都是字符串形式，  
 \* 即使传入的是数字  
 \* \return 0代表返回成功，负数代表返回错误码  
 \*/

* **常用标注**

/\*!  
 \* \bug  
 \* \todo  
 \* \var  
 \* \enum  
 \* \stuct  
 \* \class  
 \*/

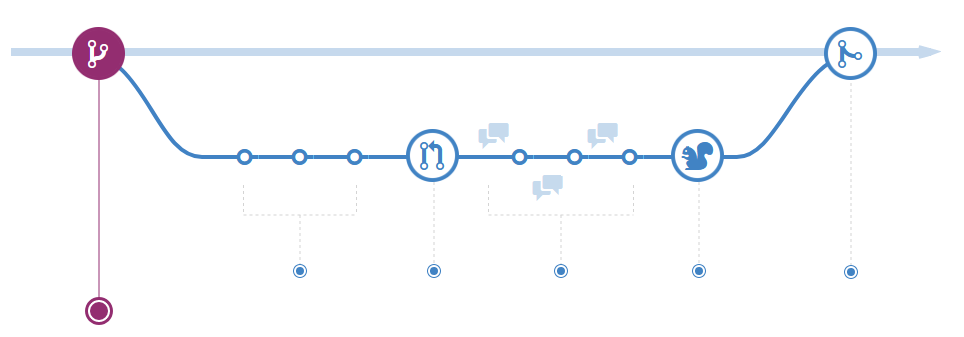
## 5. Version Control （Git or Github)

### 5.1 commit rule

* 提交要对应修改。每次提交尽量只对应一个功能改动或bug修复。
* 不提交不完整的改动；
* 确保提交质量。要有简短说明修改内容。在详细说明中写清楚改动理由和具体改了哪些内容。

### 5.2 Github flow

https://guides.github.com/introduction/flow/



主要分成以下几步：

* **create branch**: master主分支用来做产品发布，需要添加新功能或测试需要新建branch
* **add commit**：每次修改一个小功能或者fix一个小bug后都要commit，防止出现过多改动的commit
* **pull request**：功能改好后，pull request 合并分支到master上
* **discuss & code review**：审核代码，确定没有问题
* **merge**：合并到主分支

### 5.3 Git flow

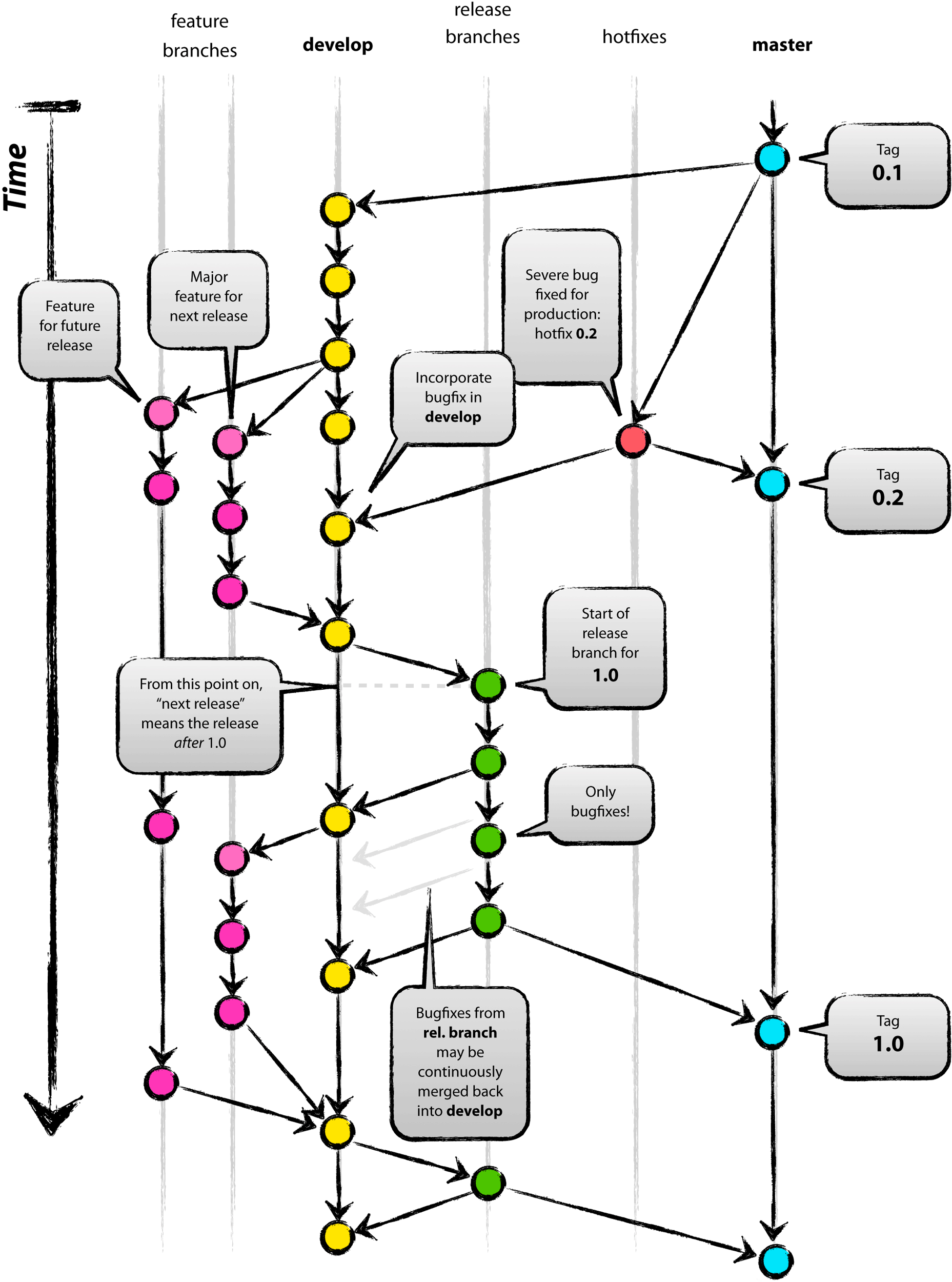
**master** 分支为最终release分支；

**develop** 分支为开发主分支；

**feature branches** 不同开发人员各自根据需要添加的功能创建分支，功能添加完成后合并到develop分支；

**release** 分支用来做release, release过程的修改需要merge到master分支和develop分支

**hotfix** 分支用来做已经release版本的紧急bug修复，bug修复好后需合并到master和develop分支；



### 5.4 Commit emoji

提交commit的时候采用下面常用的emoji，让别人直观地看到改动属于哪一类别。

🐛 :bug: for bug fix

✏️ :pencil2: 修复拼写错误

📝 :memo: 撰写文档

🔨 :hammer: 重大重构

🔥 :fire: 移除代码或文件

🎉 :tada: 初次提交

🆕 :new: 添加新特性、新功能

🚧 :construction: 工作进行中

🎨:art: 改进代码结构、代码格式

🎬 :clapper: 演示版本

💄 :lipstick: 更新UI和样式

🚑 :ambulance: 重要补丁

🔖 :bookmark: 发行、版本标签

🔀 :twisted\_rightwards\_arrows: 合并分支

## 参考文献

linux kernel C coding style

https://www.kernel.org/doc/html/v4.10/process/coding-style.html

gnome C coding style

https://developer.gnome.org/programming-guidelines/stable/c-coding-style.html.en

https://developer.gnome.org/programming-guidelines/stable/writing-good-code.html.en

gnu c coding style

https://www.gnu.org/prep/standards/html\_node/Writing-C.html

AT&T c coding style

https://www.csd.uoc.gr/~zakkak/hy255/reading/c*coding*style.pdf

华为C编程规范

https://wenku.baidu.com/view/13299a728762caaedc33d490.html#