代码风格和命名规范

代码风格

> 代码风格无好坏，目的在于统一

遵循ROS2风格，`.clang-format`文件如下：

```
# https://github.com/ament/ament_lint/blob/rolling/ament_clang_format/ament_clang_format/configuration/.clang-format
---
Language: Cpp
BasedOnStyle: Google

AccessModifierOffset: -2
AlignAfterOpenBracket: AlwaysBreak
BraceWrapping:
  AfterClass: true
  AfterFunction: true
  AfterNamespace: true
  AfterStruct: true
  AfterEnum: true
BreakBeforeBraces: Custom
ColumnLimit: 100
ConstructorInitializerIndentWidth: 0
ContinuationIndentWidth: 2
DerivePointerAlignment: false
PointerAlignment: Middle
ReflowComments: false
...
```

推荐在`.vscode/settings.json`中开启自动格式化：

```
{
    "editor.formatOnSave": true,
    // 关闭c语言的自动格式化，防止对cubemx生成的代码进行更改
    "[c]": {
```

```
        "editor.formatOnSave": false,
    },
    // 其它设置...
}
```

命名规范

> 这里还包括一些编程习惯

文件

- 文件夹：全小写+下划线，例如：`sp_engineer`
- 声明（declaration）和定义（definition）应该分离，`template` 和 `inline` 除外
- 源文件：全小写+下划线，例如：`buzzer_task.cpp`
- 头文件：
  - 全小写+下划线，`.hpp` 后缀，例如：`buzzer_task.hpp`
  - 头文件要使用 `#ifndef` 保护，防止重复引用：

```
#ifndef BUZZER_TASK_HPP
#define BUZZER_TASK_HPP

// 你的代码

#endif // BUZZER_TASK_HPP
```

变量

- 全小写+下划线（snake_case），例如：`gimbal_yaw_motor_angle`
- 非必要不缩写：未来的你会感谢你自己
- 如果缩写应该至少使用三个字母，例如：`encoder` 缩写为 `ecd`
- 有物理含义的数值应符合国际单位制：m, m/s, rad, rad/s, N, $N \cdot m$, A, V, ...
- 如果不是国际单位制，变量名应包含其单位，例如：`speed_rpm`，或者 `angle_deg`
- 非必要不使用全局变量

常量

- 运行时（runtime）常量
  - 即 `const`
  - 命名方式同普通变量
- 编译时（compile-time）常量
  - 全大写+下划线（UPPER_CASE），例如：`MAX_IOUT`
  - 应使用 `constexpr` 而不是 `#define`：

```
// correct
constexpr double PI = 3.14159265357;
// wrong
#define PI 3.14159265357
```

函数

- 全小写+下划线，例如：`set_angle(float angle)`
- 函数应该是“时不变”的：输出结果仅取决于输入参数

类

- 类名：
  - 优先首字母大写+无下划线（CamelCase），例如：`DaMiaoMotor`，
  - 缩写时，使用下划线分割，例如：`DM_Motor`
- private属性
  - 全小写+下划线，结尾加下划线，例如：`ecd_`
  - 私有属性应尽量少用，提高“时不变性”
- public属性
  - 命名同普通变量
  - 类内应使用 `this->` 强调

- 方法(method)命名方式同函数
- 谨慎使用继承，尤其避免“子类的子类”，我们的业务逻辑不应该如此复杂

命名空间

- 禁止 `using namespace ...`，类似 `std::chrono_literal` 除外
- 命名方式：全小写+下划线，例如： `io::Buzzer`，或者 `math_tools::PI`

参考

- ROS2风格：http://docs.ros.org/en/rolling/The-ROS2-Project/Contributing/Code-Style-Language-Versions.html#id1
- 上交风格：https://sjtu-robomaster-team.github.io/cpp-style-guide/