Pixhawk 原生固件 PX4 之添加 uORB 主题

本说明针对 Firmware v1.5.4

1. 添加流程说明

(1) 在 Firmware/msg 下新建 uORB 的成员变量，eg: xxx.msg

(2) 在 Firmware/msg/CMakeLists.txt 中添加话题的 xxx.msg ，作为 cmake 的编译索引。

2. 添加话题示例

（1）这里，在 Firmware/msg 下新建 fantasy.msg。内容为：

```
1  uint64 hehe1
2  uint64 hehe2
3  uint64 hehe3
4  uint64 hehe4
5
6  # TOPICS fantasy huihui jxf
```

(2) Firmware/msg 中的 CMakeLists.txt 中添加： fantasy.msg

```
  uavcan_parameter_value.msg
  ulog_stream.msg
  ulog_stream_ack.msg
  vehicle_attitude.msg
  vehicle_attitude_setpoint.msg
  vehicle_command_ack.msg
  vehicle_command.msg
  vehicle_control_mode.msg
  vehicle_force_setpoint.msg
  vehicle_global_position.msg
  vehicle_global_velocity_setpoint.msg
  vehicle_gps_position.msg
  vehicle_land_detected.msg
  vehicle_local_position.msg
  vehicle_local_position_setpoint.msg
  vehicle_rates_setpoint.msg
  vehicle_status.msg
  vision_position_estimate.msg
  vtol_vehicle_status.msg
  wind_estimate.msg
  vehicle_roi.msg
  mount_orientation.msg
  collision_report.msg
  low_stack.msg
  ca_trajectory.msg
+ fantasy.msg
  )
```

注意：

TOPICS # 号和 TOPICS 中间有一个空格。

一个消息定义就可以用于多个独立的主题。

3. 原理说明

xxx.msg 为成员；

TOPICS 为话题的定义；

在编译的时候，通过 **genmsg_cpp** 自动生成一定结构的代码，再通过 CMakeLists.txt 进行编译，所以在编译一遍后，才能具体看到所定义的话题成员。

上面的 fantasy.msg 编译后生成的 /uORB/topics/fantasy.h 文件主要内容如下：

```
struct __EXPORT fantasy_s {
#else
struct fantasy_s {
#endif
    uint64_t timestamp; // required for logger
    uint64_t hehe1;
    uint64_t hehe2;
    uint64_t hehe3;
    uint64_t hehe4;

#ifdef __cplusplus

#endif
};

/* register this as object request broker structure */
ORB_DECLARE(fantasy);
ORB_DECLARE(huihui);
ORB_DECLARE(jxf);
```


注意：

每一个生成的 C/C++结构体中，会多出一个 uint64_t timestamp 字段。这个变量作为话题运行的时间戳，用于将消息记录到日志当中。

常有人说 uORB 文件夹下没有 topics 文件夹。确实 Firmware/src/modules/uORB 目录下是没有 topics 文件夹的。但是如果是 FMUv2 或 FMUv3 系列的飞控板，关注的是编译得到的目录 Firmware/build_px4fmu-v2_default，这里面有你需要的东西。

4. 相关函数说明

主要是关于 uORB 函数的简单介绍。

- 公告

```
1  orb_advert_t orb_advertise(const struct orb_metadata *meta, const void *data)
```

- 发布

```
1  int orb_publish(const struct orb_metadata *meta, orb_advert_t hand
   le, const void *data)
```

- 订阅

```
1  int orb_subscribe(const struct orb_metadata *meta)
```

- 复制

```
1  int orb_copy(const struct orb_metadata *meta, int handle, void *b
   uffer)
```

参数说明：

orb_advert_t ：空指针 handle

const struct orb_metadata *meta ：话题 ID

const void *data ：相关数据类型指针

话题之间的发布订阅依赖于 handle 进行相关性的传递，话题的 ID 和结构通过#TOPICS fantasy 来定义；

注意：在公告和发布时用的是 handle 指针，订阅和复制用的是整形

5. 发布订阅示例

示例在 px4_simple_app.c 上进行测试

```
1  /**
2  *@file px4_simple_app.c
3  *Minimal application example for PX4 autopilot
4  *
5  *@author Fantasy <fantasyjxf@gmail.com>
6  */
7
8  #include <px4_config.h>
9  #include <px4_tasks.h>
10 #include <px4_posix.h>
```

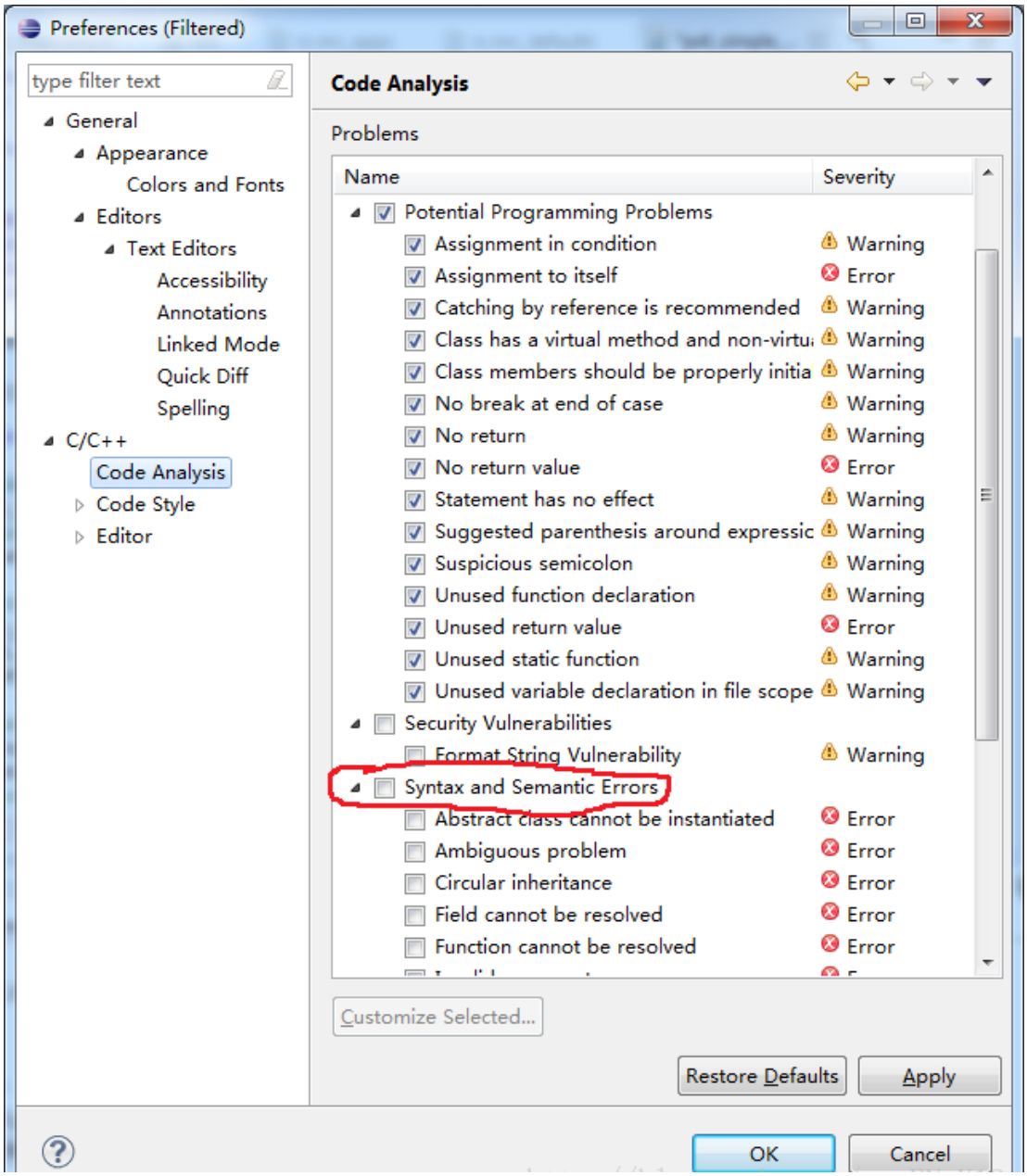
```
11 #include <unistd.h>
12 #include <stdio.h>
13 #include <poll.h>
14 #include <string.h>
15 #include <math.h>
16
17 #include <uORB/uORB.h>
18 #include <uORB/topics/sensor_combined.h>
19 #include <uORB/topics/vehicle_attitude.h>
20 #include <uORB/topics/fantasy.h>
21
22 __EXPORT int px4_simple_app_main(int argc, char *argv[]);
23
24 int px4_simple_app_main(int argc, char *argv[])
25 {
26   PX4_INFO("Hello Fantasy!");
27
28   /*定义话题结构*/
29   struct fantasy_s test;
30   /*初始化数据*/
31   memset(&test, 0, sizeof(test));
32
33   /*公告主题*/
34   /*test_pub 为 handle 指针*/
```

```
35    orb_advert_t  test_pub  =  orb_advertise(ORB_ID(fantasy),  &test);
36
37    /*test 数据赋值*/
38    test.hehe1  =  2;
39    test.hehe2  =  3;
40    test.hehe3  =  3;
41
42    /*发布测试数据*/
43    orb_publish(ORB_ID(fantasy),  test_pub,  &test);
44
45    /*订阅数据，在 copy 之前，必须要订阅*/
46    /*test_sub_fd 为 handle*/
47    int  test_sub_fd  =  orb_subscribe(ORB_ID(fantasy));
48
49    struct  fantasy_s  data_copy;
50
51    /*copy 数据*/
52    orb_copy(ORB_ID(fantasy),  test_sub_fd,  &data_copy);
53
54    /*打印*/
55    PX4_WARN("[px4_simple_app]  GanTA:\t%8.4f\t%8.4f\t%8.4f",
56                                (double)data_copy.hehe1,
57                                (double)data_copy.hehe2,
58                                 (double)data_copy.hehe3);
```

```
59
60      return 0;
61      }
```

测试效果：

最后赠送一个可能解决 Eclipse 环境下代码中到处都是奇怪警告、错误的方法



我反正是把这一栏取消勾选了，上面一栏可以留下的，代码问题分析。