# debug典型案例分析 20220509

## bug基本描述

- 出现时间：晚饭后
- 现象描述：开机后无法正常启动，上下开发板均不播放开机音乐；重新给云台板烧录代码后开机音乐响，随后能正常操控。

## debug过程

### 第一阶段

- 出发点：通过经验分析。通常该类问题可能是由于can自动重发导致的
- debug方式：控制变量：usb转can有无，两块板是否有自动重发等
- 结果：无。所有情况下均出现bug。

### 第二阶段

- 出发点：该程序bug数有限，基本框架能正常执行
- debug过程
    1. 确定大致范围：假设最底层无问题，依次单独注释各个线程的执行代码并重新上电，发现在关闭看门狗monitor线程时机器人能正常启动，关闭其他线程时则仍是bug现象，确认该bug与看门狗有关，大概率为回调函数。
    2. 找到复现方式：使用一个外置的C板，接电，将其用于板间通信的can连接。则其余部分均掉线，看门狗的各回调函数稳定触发。在这种情况下，有可能能对其使用调试软件debug
    3. 使用ozone调试，hardfault error时运行到的地址值超出该型号stm32最大内存。因此判断有指针异常。但暂时无法找出对应函数。使用print输出可知是monitor队列的第六个回调函数存在异常，但暂时并不能通过此方法定位具体函数
    4. 对call_stack窗口进行分析，产生hardfault error时正在执行的函数为Shoot_Update函数
    5. 综合"看门狗""指针超范围""Shoot_Update"等条件，判断shoot模块中有看门狗的回调函数没有正常注册
    6. 范围足够小，直接找到shoot模块各子模块lost_callback注册的位置逐个进行检查

## 成因分析

```
@ -47,8 +47,8 @@ Shoot *Shoot_Create(void) {
        friction_b_config.motor_pid_model = SPEED_LOOP;
        friction_b_config.position_fdb_model = MOTOR_FDB;
        friction_b_config.speed_fdb_model = MOTOR_FDB;
-       friction_a_config.output_model = MOTOR_OUTPUT_NORMAL;
-       friction_a_config.lost_callback = shoot_motor_lost;
+       friction_b_config.output_model = MOTOR_OUTPUT_NORMAL;
+       friction_b_config.lost_callback = shoot_motor_lost;
        PID_SetConfig(&friction_b_config.config_position, 2, 0, 0, 0, 5000);
        PID_SetConfig(&friction_b_config.config_speed, 4, 0.015, 0.8, 2000, 5000);
        obj->friction_b = Can_Motor_Create(&friction_b_config);
```

1. 注册电机时由于复制粘贴未及时修改变量名，friction_b_config中的函数指针lost_callback未赋值，会产生未正确使用的指针（野指针）
2. memset0时有遗漏。翻代码可知monitor.c中对新建的monitor，malloc得到的空间未进行memset0操作；创建电机时新建的config也未进行过memset0。因此该指针为野指针，指向未知。若set0执行则其指向NULL，被调用时会不执行直接返回而不是产生hardfault error，这也是使得该bug产生实际作用的因素之一

# 修复

将变量名进行修复。随后不改变其他条件，bug未复现。

# 操作难点

该bug的现象导致其很难通过debug复现，因为上电后debug实际上也相当于重新烧录了代码，机器人会正常启动。因此容易找不到入手点。