基于PX41.15.0固件下6X飞控的垂起硬件在环pwm_out输出验证实验（仅限完整版及以上版本）

1. 实验目的

在该固件下进行垂直起降飞机的硬件在环仿真测试，可通过本例程实现硬件在环时的真实PWM输出，从而同步控制真实电机（舵机），实现虚实结合的实验效果。

2. 实验要求

- 软件要求：Windows 10及以上版本；RflySim工具链[1]，平台安装时的编译命令为：px4_fmu-v6x_default，推荐PX4固件版本为：1.12.3。
- 硬件要求：笔记本/台式电脑1台[2]，Pixhawk 6X 1台；数据线 2台，使用 Pixhawk 6X 飞控，平台安装时的编译命令为：px4_fmu-v6x_default

3.实验地址

例程目录：[安装目录]\RflySimAPIs\5.RflySimFlyCtrl\0.ApiExps\19.VTOL_PWM_out_Test

- StandardVtolModel_HITL.bat：硬件在环仿真批处理文件
- qgcparam.params：QGC参数文件
- StandardVtolModel.dll：垂起无人机动力学模型动态链接库
- px4_fmu-v6x_default.px4：支持该功能的PX4固件

4. 实验内容或步骤

这是在V6X基础上修改编译的1.15.0固件，可以实现硬件在环仿真时（真机机架）的PWM硬件输出，适用但不仅限于垂起飞机。

4.1.步骤1：连接飞控

硬件在环仿真需要准备一个飞控，将飞控通过 USB
线连接电脑，并确保完成硬件在环仿真配置。注意，本图使用Pixhawk6x飞控，其他飞控配置方法类似（推荐使用Pixhawk飞控）。

4.2.步骤2：上传固件文件

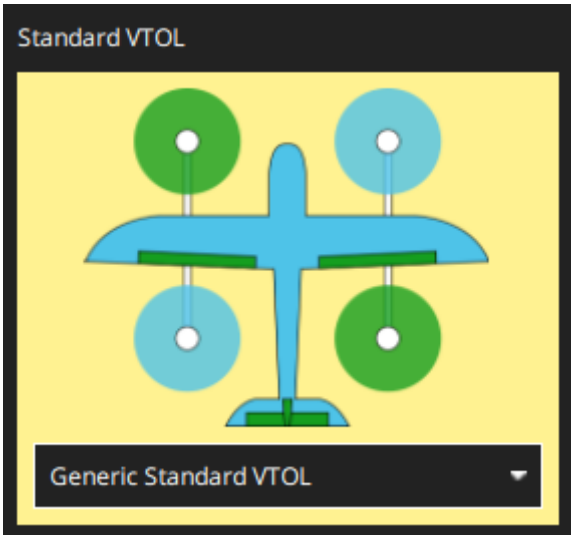
如果对RflySim平台飞控固件上传有疑问，可以扫描下方二维码，观看教程。

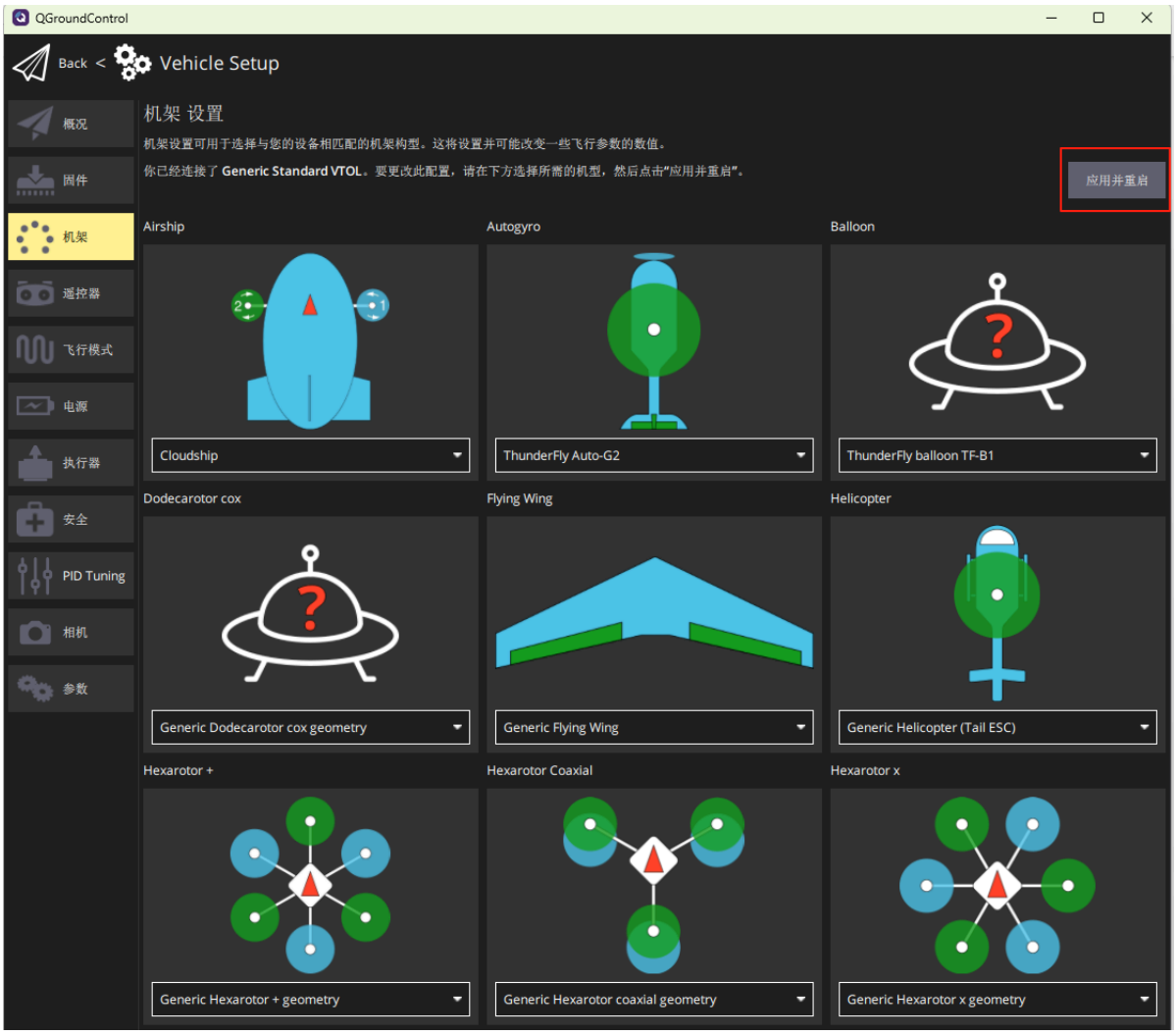
在 Rflytools 文件夹中打开QGC 地面站：

点击“Firmware”（固件）标签，此时用 USB 数据线连接 Pixhawk 自驾仪，地面站会自动检测自驾仪。勾选“advanced settings”（高级设置）选框；点击“Standard Version (stable)”（标准版 Stable）标签 – “Custom firmware file ...”（自定义固件文件...）选项，再点“确定”：

4.3.步骤3：设置硬件在环机架

进入 Vehicle Setup 页面，找到的机架，在机架界面设置机架型号为“Generic Standard VTOL”，设置完毕后点击右侧上方“应用并重启”



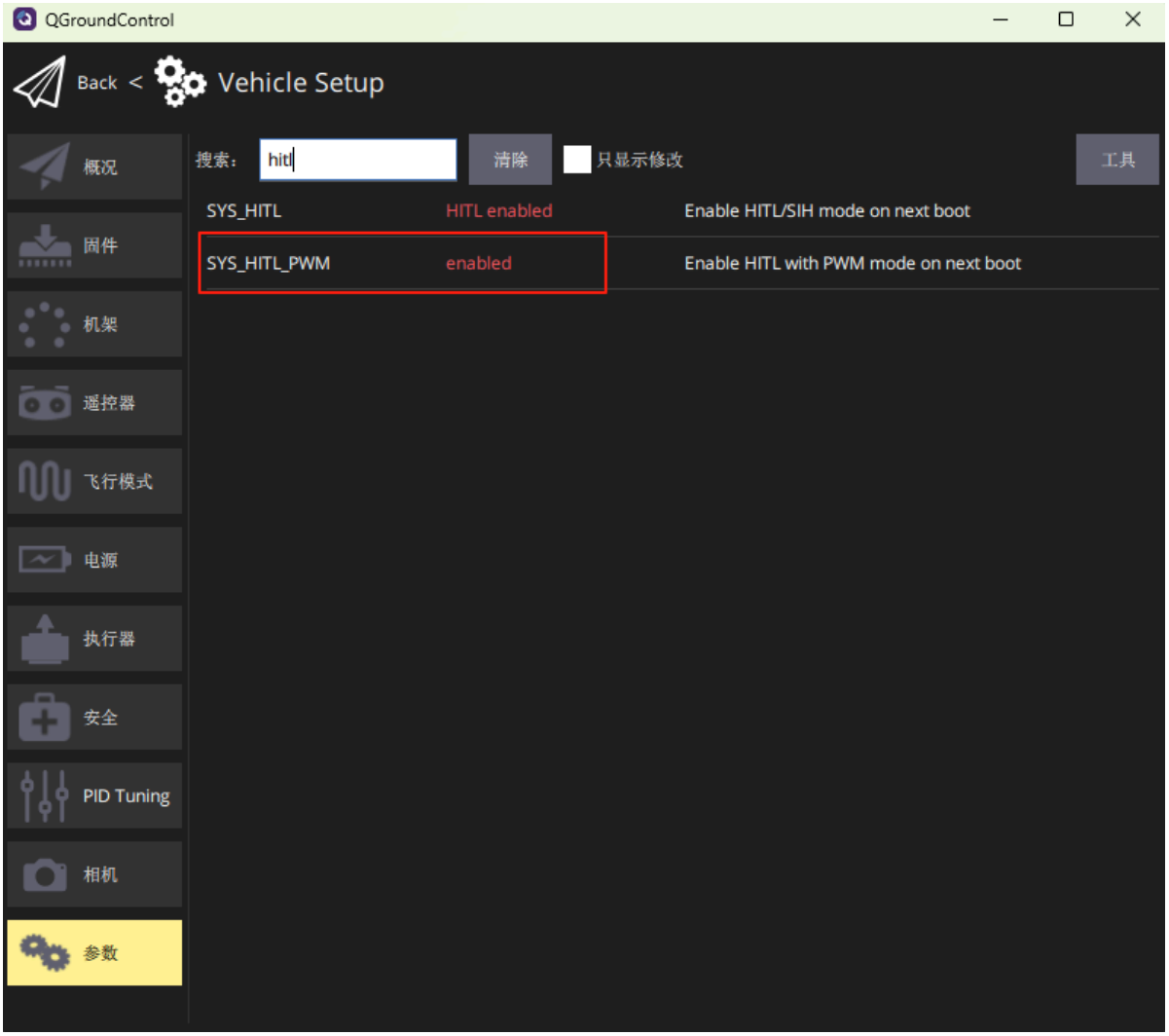


注：本固件执行器输出为AUX，对应的是飞控中FMU PWM OUT。

FMU PWM OUT (AUX OUT)

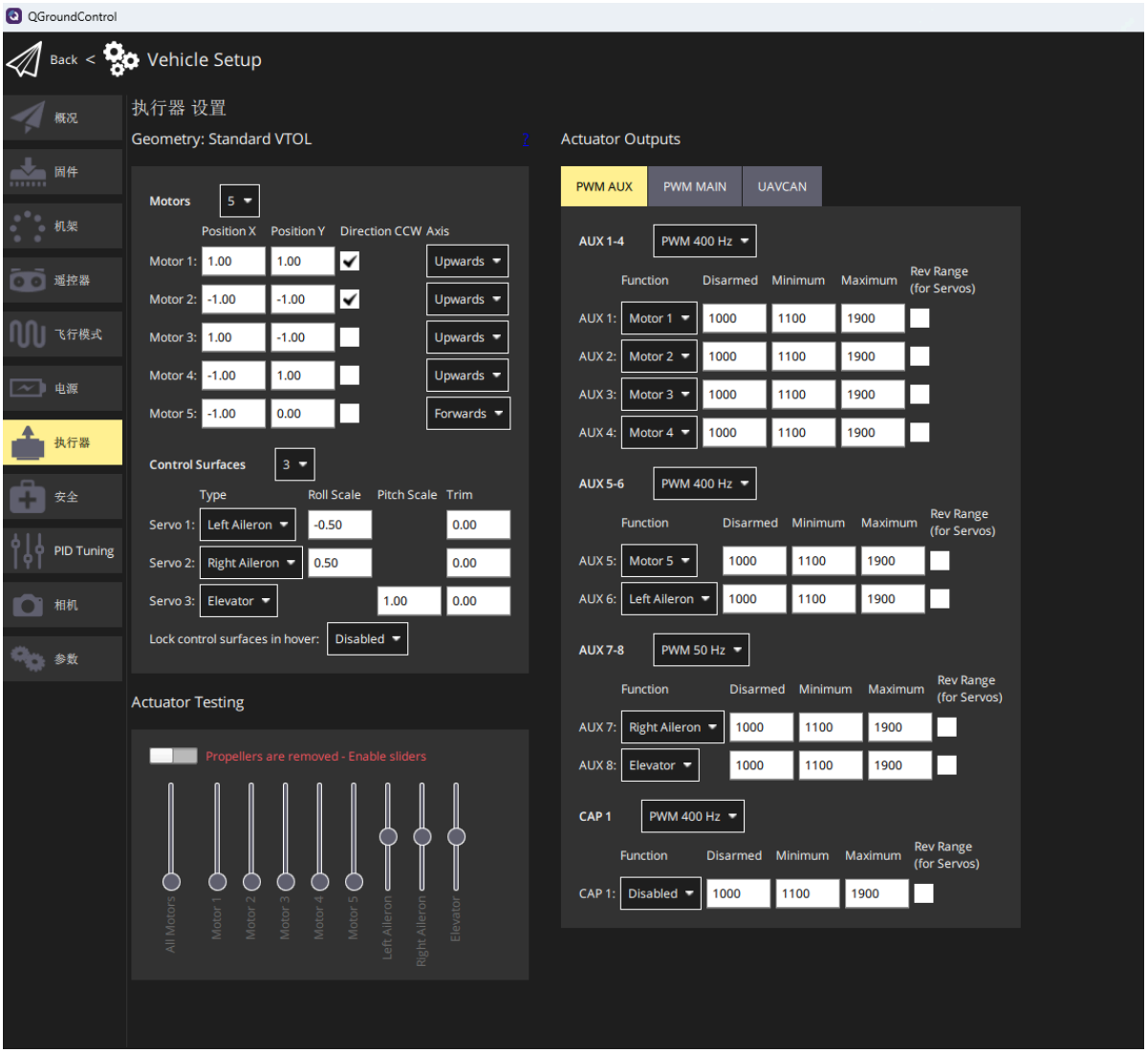
Pin	Signal	Voltage
1 (Red)	VDD_Servo	
2 (Black)	FMU_CH1	+3.3V
3 (Black)	FMU_CH2	+3.3V
4 (Black)	FMU_CH3	+3.3V
5 (Black)	FMU_CH4	+3.3V
6 (Black)	FMU_CH5	+3.3V
7 (Black)	FMU_CH6	+3.3V
8 (Black)	FMU_CH7	+3.3V
9 (Black)	FMU_CH8	+3.3V
10 (Black)	GND	GND

加了新参数SYS_HITL_PWM，并设置为enable，否则执行器部分报错，PWM AUX部分不能正常显示启用，更不能配置执行器参数。



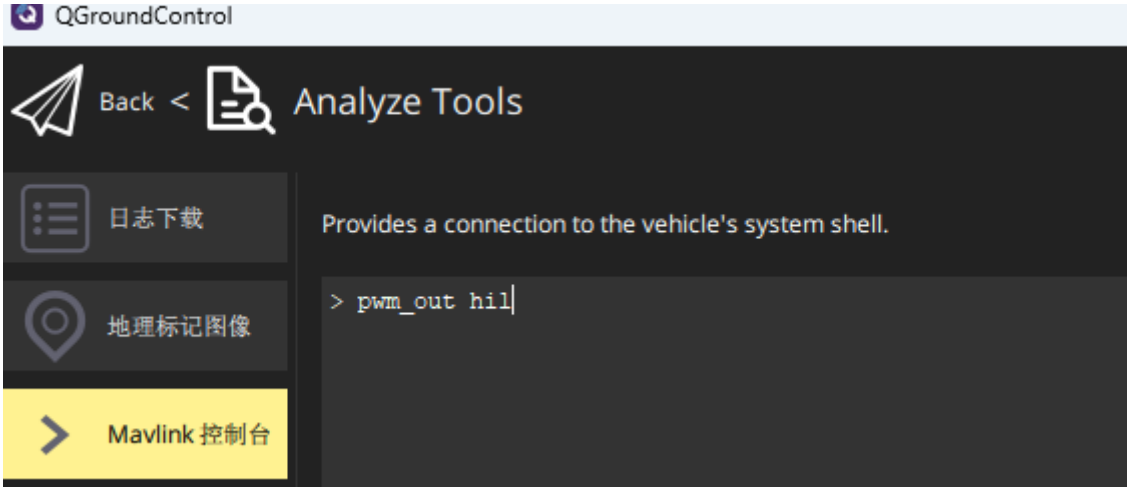
对应的PWM

AUX执行器配置如下可参照设置（与PX4.IO中该机架的执行器设置不同，以平台中动力学模型为主），参数文件qgcparam.params也可直接通过QGC加载。



注： 此处的映射需要与仿真模型中使用的动力学模型定义的通道顺序一致。

在QGC的Mavlink控制台输入以下命令pwm_out hil并回车，即可进行硬件在环仿真。

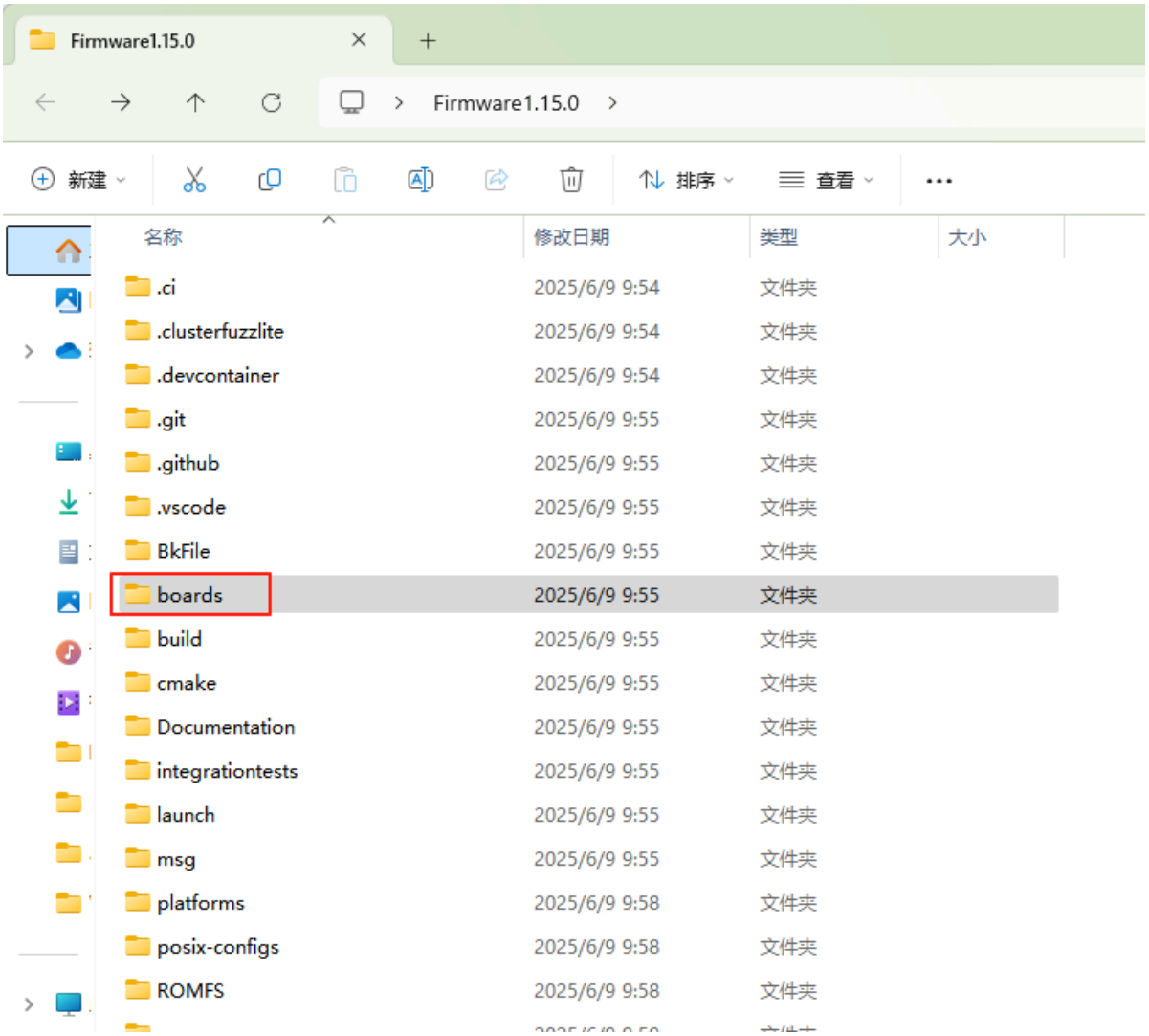


之后测试步骤与软件在环仿真的步骤相同，初始化完成后，上传航线任务并起飞，观察无人机能否按照期望的轨迹航行。

5. 关键知识点

步骤5.1 启动文件配置

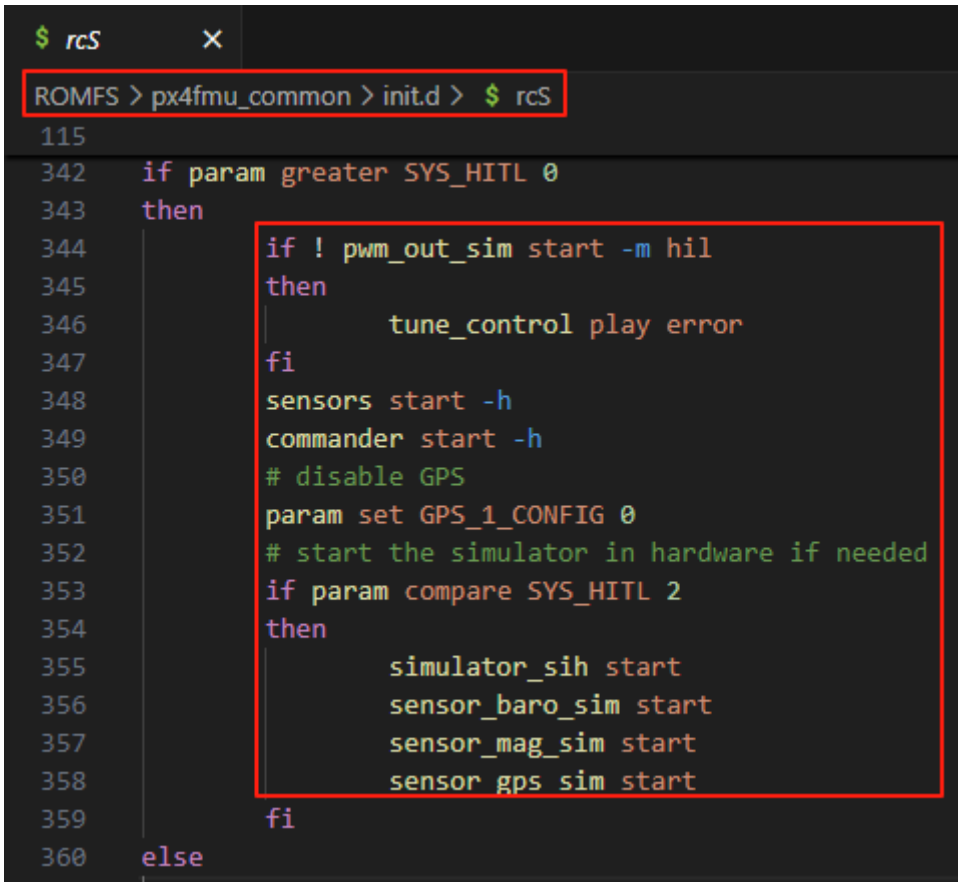
建议在Firmware文件夹下备份boards文件夹，保留原有代码的完整性，避免直接改动产生影响。



阅读rcS启动文件，明确启动了哪些参数文件。在rcS启动文件中，定义了一个新的启动模式（当SYS_HITL_PWM>0时启用触发）。

注：rcS文件路径：

【RflySim安装路径】\Firmware\ROMFS\px4fmu_common\init.d\rcS




```
if ! pwm_out_sim start -m hil
then
    tune_control play error
fi
sensors start -h
commander start -h
# disable GPS
param set GPS_1_CONFIG 0
# start the simulator in hardware if needed
if param compare SYS_HITL 2
then
    simulator_sih start
    sensor_baro_sim start
    sensor_mag_sim start
    sensor_gps_sim start
fi
```

新模式会加载必要的传感器驱动和配置文件，为同时进行HITL仿真和真实PWM输出做准备。rcS启动文件增加以下模式，以及启动硬件在环和PWM硬件输出需要的所有传感器。

ROMFS > px4fmu_common > init.d > $ rcS

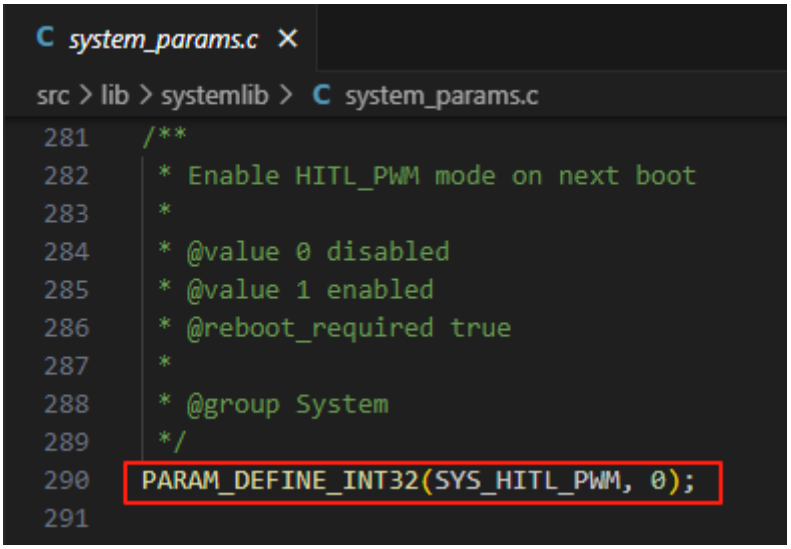
```
115   else
341           #
342           if param greater SYS_HITL 0
343           then
344                   if ! pwm_out_sim start -m hil
345                   then
346                           tune_control play error
347                   fi
348
349                   sensors start -h
350                   commander start -h
351                   # disable GPS
352                   param set GPS_1_CONFIG 0
353
354                   # start the simulator in hardware if needed
355                   if param compare SYS_HITL 2
356                   then
357                           simulator_sih start
358                           sensor_baro_sim start
359                           sensor_mag_sim start
360                           sensor_gps_sim start
361                   fi
362
363           else
364                   #
365                   # board sensors: rc.sensors
366                   #
367                   set BOARD_RC_SENSORS ${R}etc/init.d/rc.board_sensors
368                   if [ -f $BOARD_RC_SENSORS ]
369                   then
370                           echo "Board sensors: ${BOARD_RC_SENSORS}"
371                           . $BOARD_RC_SENSORS
372                   fi
373                   unset BOARD_RC_SENSORS
374
375                   . ${R}etc/init.d/rc.sensors
376
377                   if param compare -s BAT1_SOURCE 2
378                   then
379                           esc_battery start
380                   fi
381
382                   if ! param compare BAT1_SOURCE 1
383                   then
384                           battery_status start
385                   fi
386
387                   sensors start
388                   commander start
389
390                   dshot start
391                   pwm_out start
392           fi
```

5.2 参数配置

在system_params.c文件中添加了新的参数SYS_HITL_PWM，该参数可以在QGroundControl (QGC)地面站中设置（Enable/Disable），搭配rcS中的新模式使用，用于控制是否在HITL模式下启用真实PWM输出功能。

注：system_params.c文件路径：

【RflySim安装路径】\Firmware\src\lib\systemlib\system_params.c



5.3 PWM输出驱动修改

以上关于启动配置的修改能保证SYS_HITL_PWM大于0时，启动我们期望的参数文件和pwm_out输出，为了能达到仿真时既有actuator_output输出，也能有pwm_out输出，后面需要修改pwm_out.cpp和pwm_out.hpp，目的是让pwm_out驱动在输出PWM值的同时也能支持硬件在环仿真。

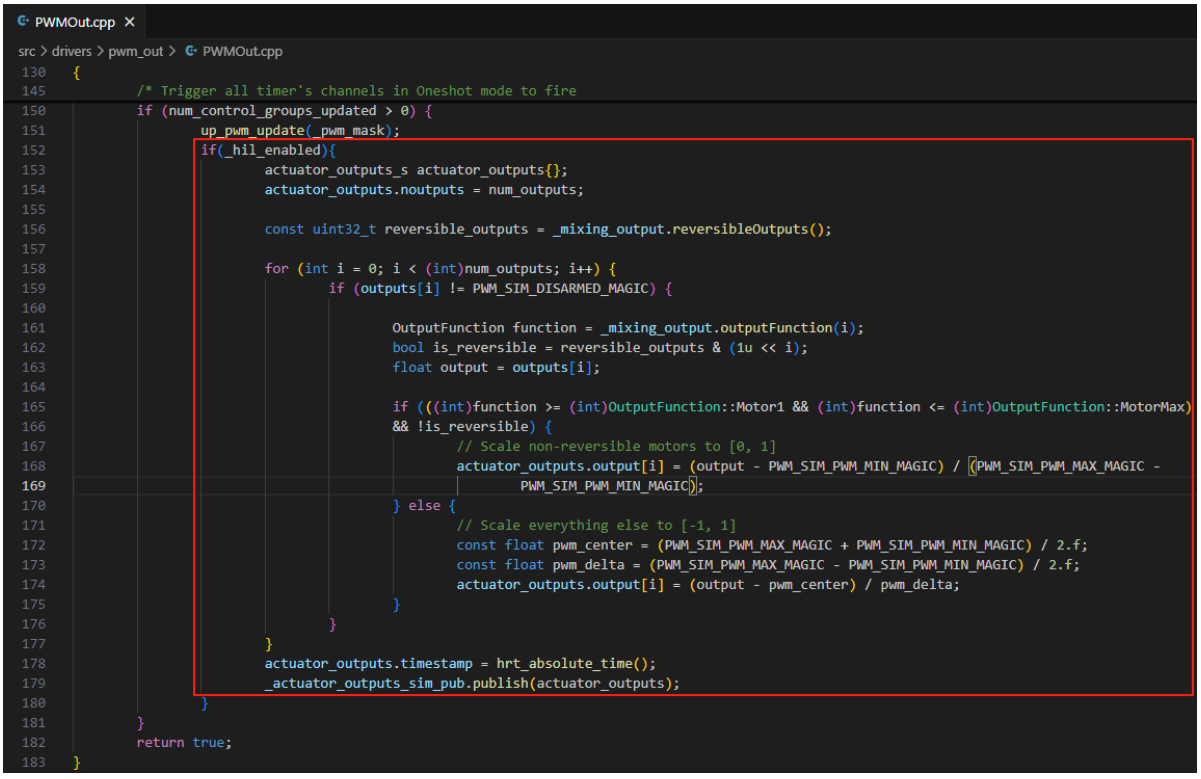
注：pwm_out.cpp和pwm_out.hpp路径：

【RflySim安装路径】\Firmware\src\drivers\pwm_out\

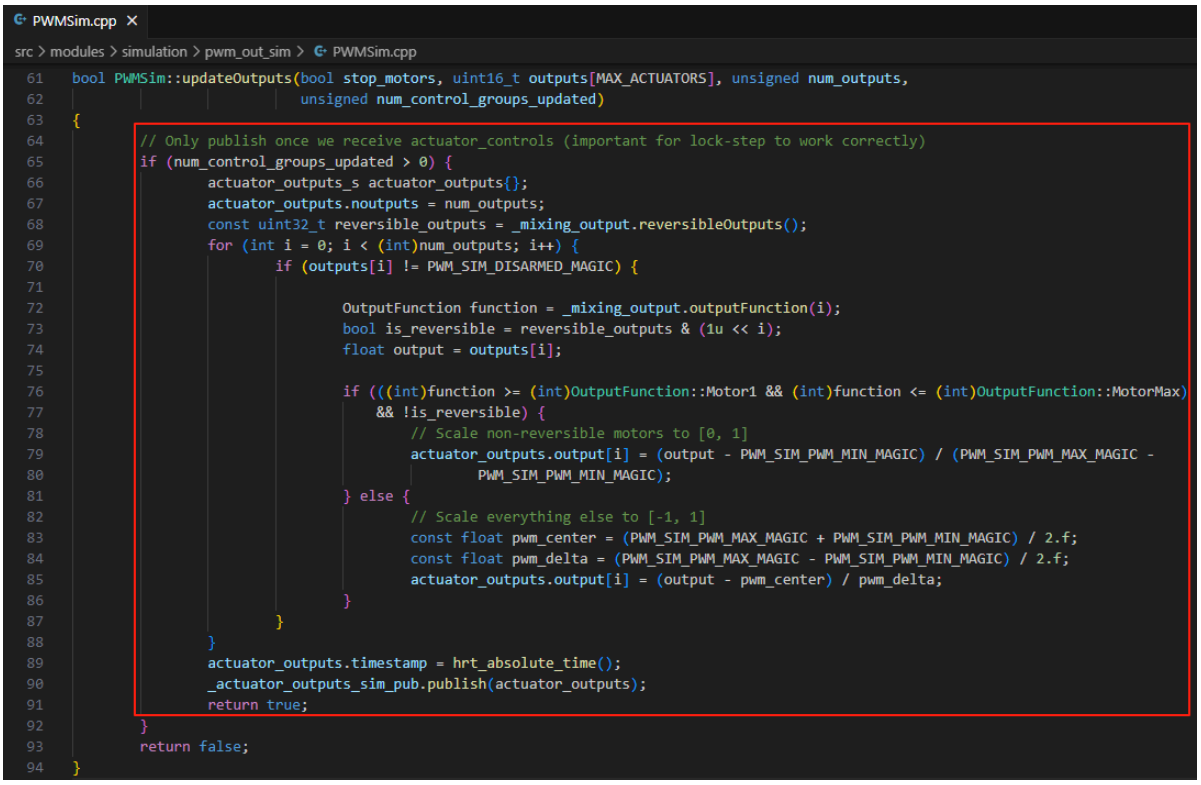
PWMSim.cpp和PWMSim.hpp路径：

【RflySim安装路径】\Firmware\src\modules\simulation\pwm_out_sim

修改1：仿照PWMSim.cpp实现模拟PWM输出




```cpp
if(_hil_enabled){
    actuator_outputs_s actuator_outputs{};
    actuator_outputs.noutputs = num_outputs;
    const uint32_t reversible_outputs = _mixing_output.reversibleOutputs();

    for (int i = 0; i < (int)num_outputs; i++) {
        if (outputs[i] != PWM_SIM_DISARMED_MAGIC) {
            OutputFunction function = _mixing_output.outputFunction(i);
            bool is_reversible = reversible_outputs & (1u << i);
            float output = outputs[i];

            if (((int)function >= (int)OutputFunction::Motor1 &&
(int)function <= (int)OutputFunction::MotorMax) && !is_reversible) {
                // Scale non-reversible motors to [0, 1]
                actuator_outputs.output[i] = (output -
PWM_SIM_PWM_MIN_MAGIC) / (PWM_SIM_PWM_MAX_MAGIC - PWM_SIM_PWM_MIN_MAGIC);
            } else {
                // Scale everything else to [-1, 1]
                const float pwm_center = (PWM_SIM_PWM_MAX_MAGIC +
PWM_SIM_PWM_MIN_MAGIC) / 2.f;
                const float pwm_delta = (PWM_SIM_PWM_MAX_MAGIC -
PWM_SIM_PWM_MIN_MAGIC) / 2.f;
                actuator_outputs.output[i] = (output - pwm_center) /
pwm_delta;
            }
        }
    }

    actuator_outputs.timestamp = hrt_absolute_time();
    _actuator_outputs_sim_pub.publish(actuator_outputs);
}
```

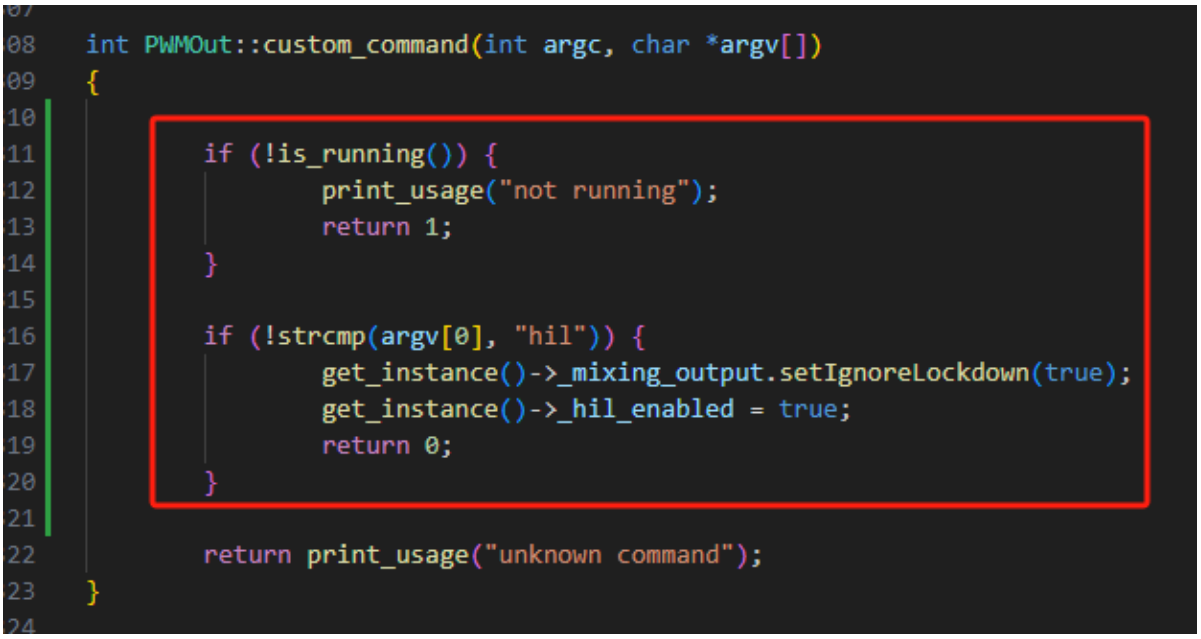



```cpp
if (num_control_groups_updated > 0) {
    actuator_outputs_s actuator_outputs{};
    actuator_outputs.noutputs = num_outputs;
    const uint32_t reversible_outputs = _mixing_output.reversibleOutputs();

    for (int i = 0; i < (int)num_outputs; i++) {
        if (outputs[i] != PWM_SIM_DISARMED_MAGIC) {
            OutputFunction function = _mixing_output.outputFunction(i);
            bool is_reversible = reversible_outputs & (1u << i);
            float output = outputs[i];

            if (((int)function >= (int)OutputFunction::Motor1 &&
(int)function <= (int)OutputFunction::MotorMax) && !is_reversible) {
                // Scale non-reversible motors to [0, 1]
                actuator_outputs.output[i] = (output -
PWM_SIM_PWM_MIN_MAGIC) / (PWM_SIM_PWM_MAX_MAGIC - PWM_SIM_PWM_MIN_MAGIC);
            } else {
                // Scale everything else to [-1, 1]
                const float pwm_center = (PWM_SIM_PWM_MAX_MAGIC +
PWM_SIM_PWM_MIN_MAGIC) / 2.f;
                const float pwm_delta = (PWM_SIM_PWM_MAX_MAGIC -
PWM_SIM_PWM_MIN_MAGIC) / 2.f;
                actuator_outputs.output[i] = (output - pwm_center) /
pwm_delta;
            }
        }
    }

    actuator_outputs.timestamp = hrt_absolute_time();
    _actuator_outputs_sim_pub.publish(actuator_outputs);
    return true;
}
```

PWM值更新 (pwm_out.cpp)，修改了PWM值更新的逻辑。当pwm_out驱动运行时，它仍然使用up_pwm_update来驱动真实的硬件PWM输出。

修改2：自定义指令

添加了一个自定义控制台命令`pwm_out hil`。执行此命令会设置一个内部标志_hil_enabled，用来激活上述修改的逻辑，同时解除lockdown限制和触发硬件在环。关键一步是调用setIgnoreLockdown(true)，因为在标准的HITL模式下，PX4会锁定（lockdown）真实的PWM输出以防止意外启动电机，这个修改解除了该限制，允许真实PWM输出。



```
int PWMOut::custom_command(int argc, char *argv[])
{

        if (!is_running()) {
                print_usage("not running");
                return 1;
        }

        if (!strcmp(argv[0], "hil")) {
                get_instance()->_mixing_output.setIgnoreLockdown(true);
                get_instance()->_hil_enabled = true;
                return 0;
        }

        return print_usage("unknown command");
}
```

```
if (!is_running()) {
    print_usage("not running");
    return 1;
}

if (!strcmp(argv[0], "hil")) {
    get_instance()->_mixing_output.setIgnoreLockdown(true);
    get_instance()->_hil_enabled = true;
    return 0;
}
```

修改3：添加指令提示信息

```cpp
int PWMOut::print_usage(const char *reason)
{
	if (reason) {
		PX4_WARN("%s\n", reason);
	}

	PRINT_MODULE_DESCRIPTION(
		R"DESCR_STR(
### Description
This module is responsible for driving the output pins. For boards without a separate IO chip
(eg. Pixracer), it uses the main channels. On boards with an IO chip (eg. Pixhawk), it uses the AUX channels, and the
px4io driver is used for main ones.

)DESCR_STR");

	PRINT_MODULE_USAGE_NAME("pwm_out", "driver");
	PRINT_MODULE_USAGE_COMMAND("start");
	PRINT_MODULE_USAGE_COMMAND("hil");
	PRINT_MODULE_USAGE_DEFAULT_COMMANDS();

	return 0;
}

extern "C" __EXPORT int pwm_out_main(int argc, char *argv[])
{
	return PWMOut::main(argc, argv);
}
```

修改4：修改头文件

照PWMSim修改PWMOut.hpp，在头文件中添加必要的成员变量和函数声明以支持上述.cpp文件的修改。

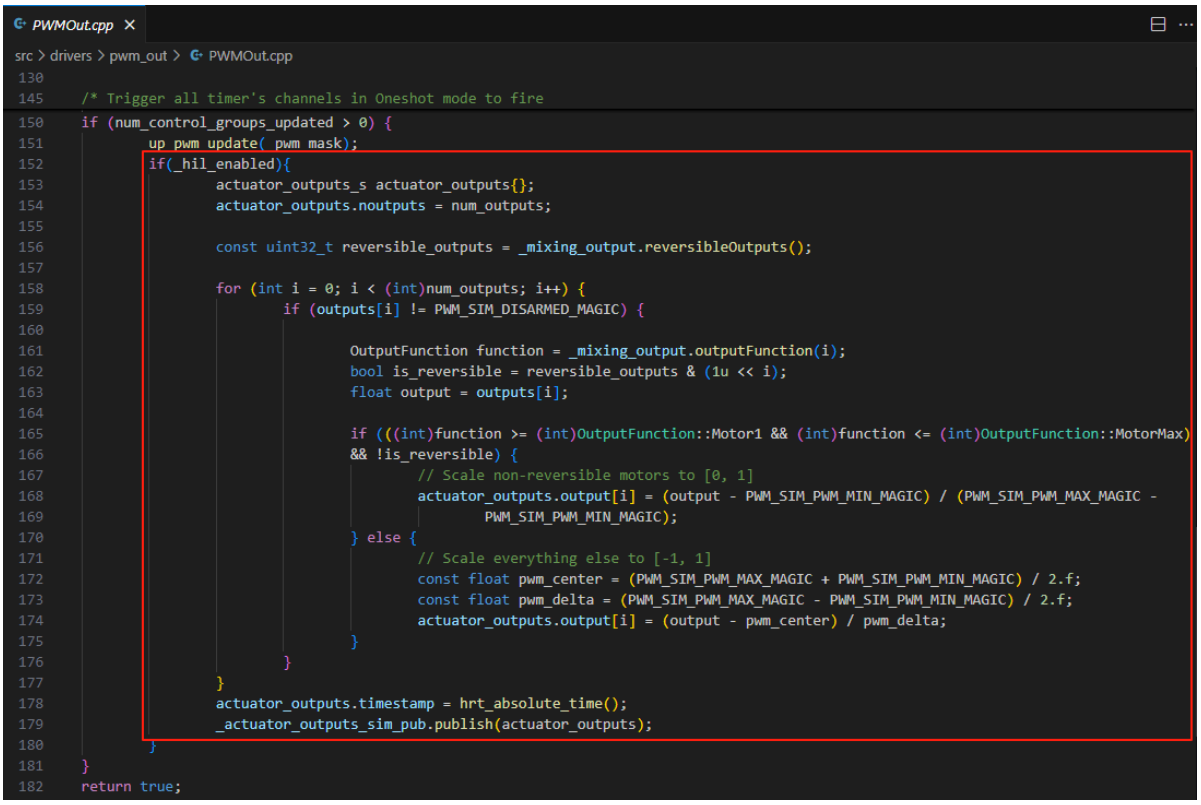
```cpp
private:
	void Run() override;

	static constexpr uint16_t PWM_SIM_DISARMED_MAGIC = 900;
	static constexpr uint16_t PWM_SIM_FAILSAFE_MAGIC = 600;
	static constexpr uint16_t PWM_SIM_PWM_MIN_MAGIC = 1000;
	static constexpr uint16_t PWM_SIM_PWM_MAX_MAGIC = 2000;
	bool _hil_enabled{false};

	void update_params();
	bool update_pwm_out_state(bool on);

	MixingOutput _mixing_output{PARAM_PREFIX, DIRECT_PWM_OUTPUT_CHANNELS, *this, MixingOutput::SchedulingPolicy::Auto, true};

	int _timer_rates[MAX_IO_TIMERS] {};

	uORB::SubscriptionInterval _parameter_update_sub{ORB_ID(parameter_update), 1_s};
	uORB::Publication<actuator_outputs_s> _actuator_outputs_sim_pub{ORB_ID(actuator_outputs_sim)};
	unsigned		_num_outputs{DIRECT_PWM_OUTPUT_CHANNELS};

	bool			_pwm_on{false};
	uint32_t		_pwm_mask{0};
	bool			_pwm_initialized{false};
	bool			_first_update_cycle{true};

	perf_counter_t	_cycle_perf{perf_alloc(PC_ELAPSED, MODULE_NAME": cycle")};
	perf_counter_t	_interval_perf{perf_alloc(PC_INTERVAL, MODULE_NAME": interval")};
};
```

5.4 总体修改思路

总体的修改思路是在不影响整体逻辑的前提下，在真机输出的基础上加上硬件在环的功能，具体的修改思路如下：

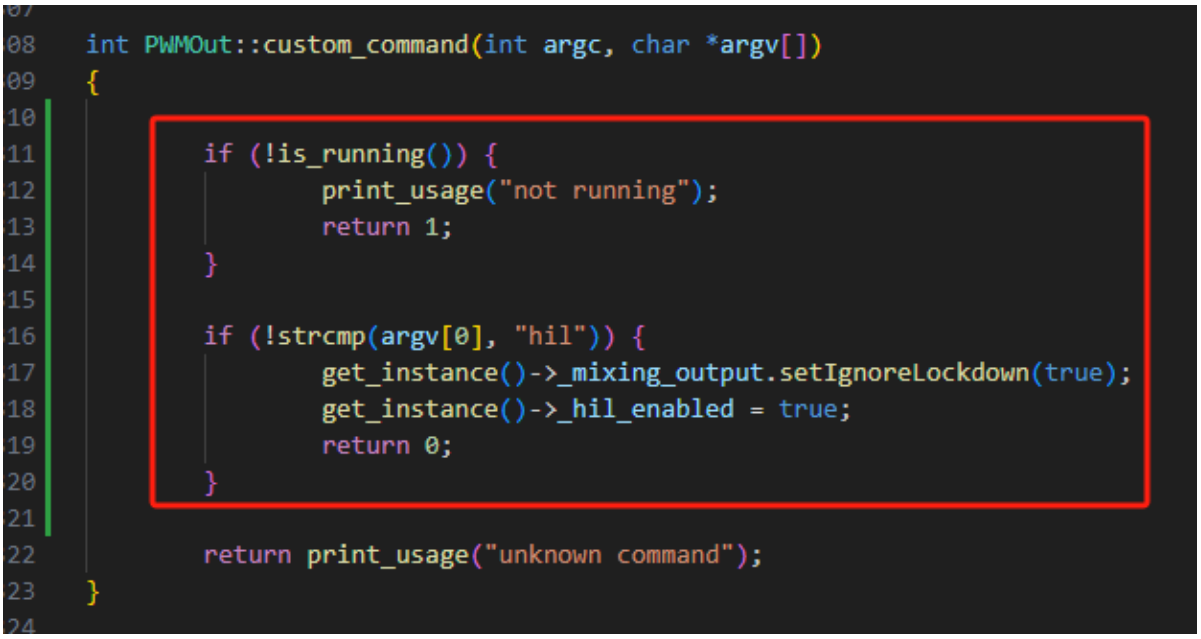
上述介绍了启动文件的内容是新参数SYS_HITL_PWM启动一个新模式，下面主要解释PWMOut部分。

PWMOut.cpp中原本只用up_pwm_update触发PWM的更新，现为了同时实现HITL发布模拟执行器的输出的要求加上标记的内容：将PWM值进行数值转换，非可逆电机映射到[0,1]，可逆电机（舵机）映射到[-1,1]，最后publish。




```
if(_hil_enabled){
    actuator_outputs_s actuator_outputs{};
    actuator_outputs.noutputs = num_outputs;
    const uint32_t reversible_outputs = _mixing_output.reversibleOutputs();

    for (int i = 0; i < (int)num_outputs; i++) {
        if (outputs[i] != PWM_SIM_DISARMED_MAGIC) {
            OutputFunction function = _mixing_output.outputFunction(i);
            bool is_reversible = reversible_outputs & (1u << i);
            float output = outputs[i];

            if (((int)function >= (int)OutputFunction::Motor1 &&
(int)function <= (int)OutputFunction::MotorMax)&& !is_reversible) {
                // Scale non-reversible motors to [0, 1]
                actuator_outputs.output[i] = (output -
PWM_SIM_PWM_MIN_MAGIC) / (PWM_SIM_PWM_MAX_MAGIC - PWM_SIM_PWM_MIN_MAGIC);
            } else {
                // Scale everything else to [-1, 1]
                const float pwm_center = (PWM_SIM_PWM_MAX_MAGIC +
PWM_SIM_PWM_MIN_MAGIC) / 2.f;
                const float pwm_delta = (PWM_SIM_PWM_MAX_MAGIC -
PWM_SIM_PWM_MIN_MAGIC) / 2.f;
                actuator_outputs.output[i] = (output - pwm_center) /
pwm_delta;
            }
        }
    }

    actuator_outputs.timestamp = hrt_absolute_time();
    _actuator_outputs_sim_pub.publish(actuator_outputs);
}
```

在调用PWMOut驱动的前提下先确保了PWM的硬件输出，加上PWMSim的模拟输出部分，在开启硬件在环时可以输出模拟控制量实现仿真实现。



```
int PWMOut::custom_command(int argc, char *argv[])
{

        if (!is_running()) {
                print_usage("not running");
                return 1;
        }

        if (!strcmp(argv[0], "hil")) {
                get_instance()->_mixing_output.setIgnoreLockdown(true);
                get_instance()->_hil_enabled = true;
                return 0;
        }

        return print_usage("unknown command");
}
```

```
if (!is_running()) {
    print_usage("not running");
    return 1;
}

if (!strcmp(argv[0], "hil")) {
    get_instance()->_mixing_output.setIgnoreLockdown(true);
    get_instance()->_hil_enabled = true;
    return 0;
}
```

这部分代码实现了关于PWMOut的一些自定义需求，当command为`pwmout hil`时，会触发标志为_hil_enbled触发前段修改，也会触发setIgnoreLockdown绕过HITL的安全限制，是硬件在环能顺利进行的必要条件。

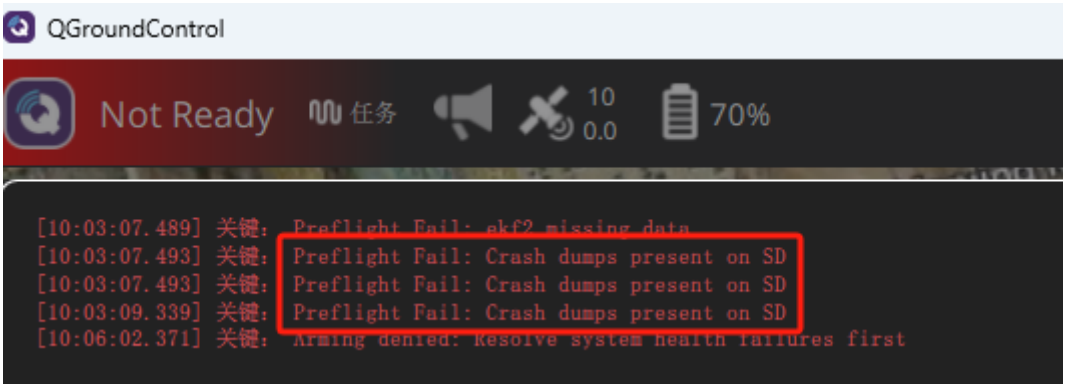
6.参考资料

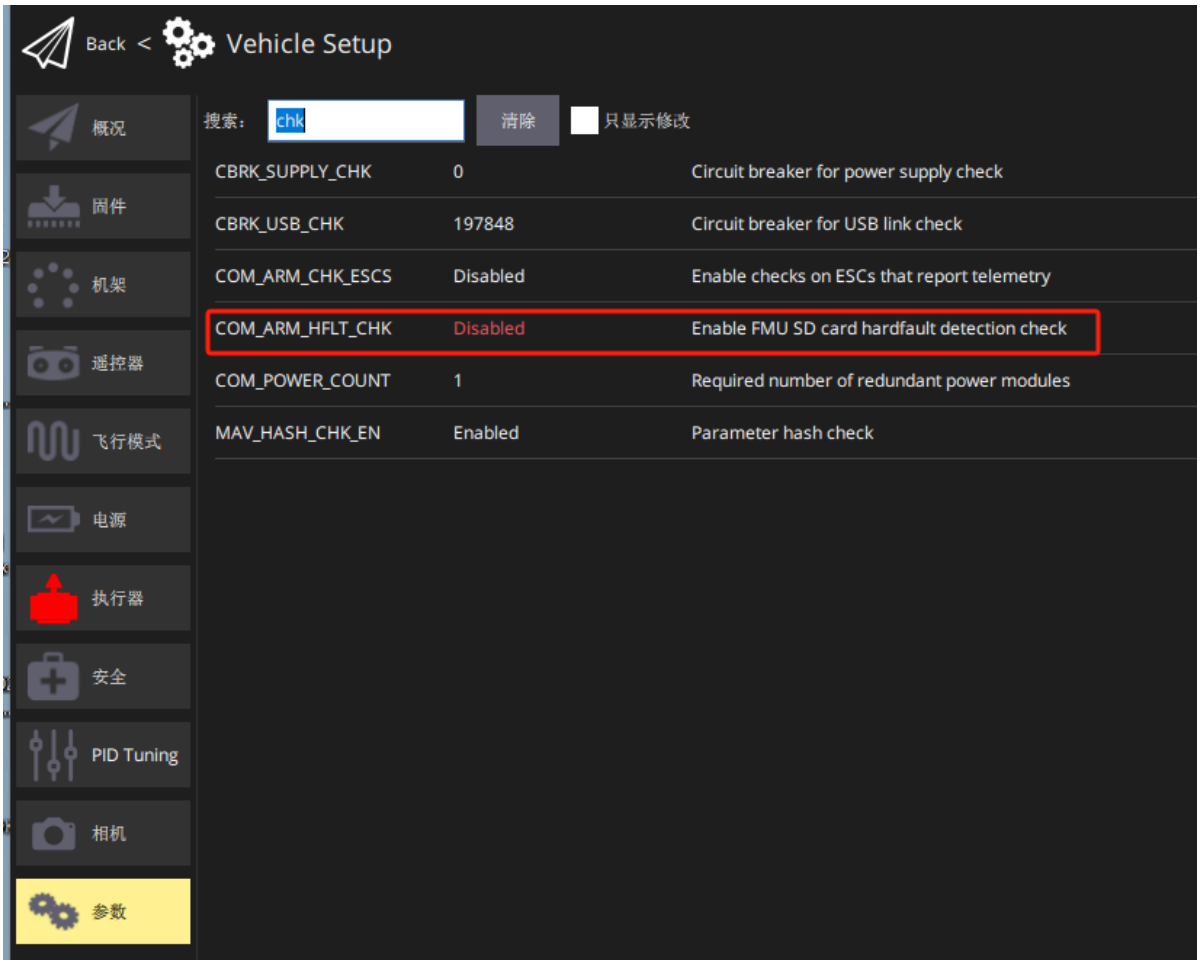
1. PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf 中 DLL/SO 模型与通信接口的重要参数部分。
2. PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf 中的环境配置

7.常见问题

Q1: 可能会遇到SD卡接收崩溃数据?

A1: 在参数中修改如下数据为Disabled即可。





---

1. https://rflysim.com/ ↩

2. 推荐配置请见：https://rflysim.com/doc/zh/HowToInstall.pdf ↩