Chapter Outline

PX4 Architecture and Control Development Interfaces

Chapter 5
Filtering, State Estimation,
and Low-Level Control

5.1 Filtering and Low-Level Control

5.2 Sensor Calibration and Estimation

5.3 Controller Design and Allocation

5.4 Control Performance and Tuning

5.5 PX4 Architecture and Interfaces

5.6 Control Development and Testing

5.7 Other Vehicles and Advanced Cases

5.8 Exercises and Classroom Discussion

4 Units in This Section

5.5.1 PX4 Autopilot and Code Architecture

Textbook: Secs. 5.2.1.1–5.2.1.2

5.5.2 Common Interfaces and Custom Development

Textbook: Sec. 5.2.3

5.5.3 Generated Controller Integration with PX4

Textbook: Secs. 5.2.1.3–5.2.1.4

5.5.4 Interface Validation and Module Replacement

Textbook: Secs. 5.2.3, 5.3.1

5.5.1 PX4 Autopilot and Code Architecture: PX4 Roles and Data Flow

The Flight Controller Links Sensing, Estimation, and Control

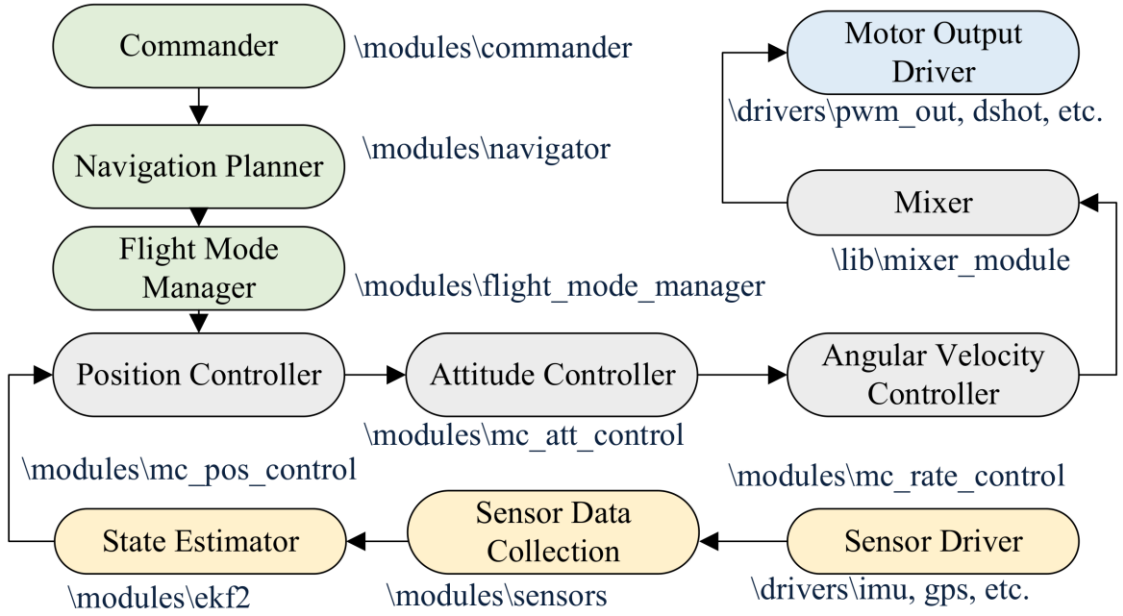


Fig. 5.5 PX4 Source Code Modules and Control Flow Diagram

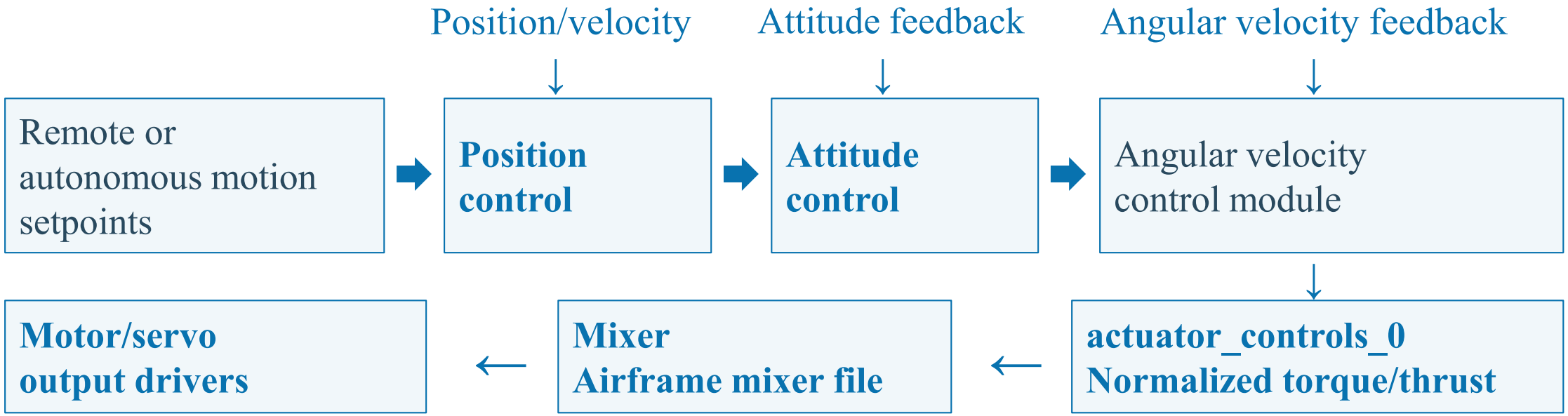
Yellow: sensor drivers → data aggregation → ekf2 state estimates; green: commands, navigation and modes set goals.

Gray: position → attitude → rate control → mixer → motor output drivers (bundled v1.12.3 path).

5.5.1 PX4 Autopilot and Code Architecture: Multicopter Control Chain in v1.12.3

Estimated States and Motion Setpoints Meet in the Controller

State feedback: sensor processing/estimation provides position, attitude and body angular velocity.



The mode selects active loops; thrust passes down the chain, while rate error produces only torque requests.

5.5.1 PX4 Autopilot and Code Architecture: PX4 Features and Tradeoffs

Modules Are Reusable; Runtime Resources Are Shared

Reusable modules: sensing, estimation, control, missions and communication.
Platforms: NuttX real-time operating system on flight controllers; Linux for simulation.

WorkItem: one scheduled computation executed on a shared work-queue thread.
In v1.12.3, mc_rate_control is a work item triggered by angular velocity updates.

Licenses: PX4 uses BSD-3 and permits closed-source redistribution; ArduPilot uses GPLv3 and requires source for distributed derivatives.

Execution	Resources	Development constraints
Dedicated task	Own stack and task priority	Limited by CPU, memory and scheduling
Work queue	Items on one queue share a thread and stack	Avoid blocking; bound each execution time

5.5.1 PX4 Autopilot and Code Architecture: Finding Source Code and Resources

Follow Data Flow to Source Code and the Student Package

Question	PX4 v1.12.3 source paths ([安装目录]/Firmware)
Fields, units and timestamps	msg/*.msg
State estimation and control algorithms	src/modules/ekf2, mc_att_control, mc_rate_control
Mixing and hardware output	src/lib/mixer, ROMFS/px4fmu_common/mixers; src/drivers
Startup and airframe configuration	ROMFS/px4fmu_common/init.d

Source: [安装目录]/Firmware (bundled default: v1.12.3); board configuration: boards/.
Example root: [安装目录]/RflySimAPIs/5.RflySimFlyCtrl; interface overview: API.pdf in that folder.
Installer version: V5.01-20260831; read each example's Readme.pdf first.

5.5.1 PX4 Autopilot and Code Architecture: Control Outputs Across Versions

Confirm the Firmware Version Before Choosing an Example

v1.12.3 (bundled default): actuator_controls_0 → mixer → output drivers

v1.14 and later: vehicle_torque_setpoint + vehicle_thrust_setpoint
→ control_allocator → actuator_motors / actuator_servos → output drivers

① Default version
Bundled PX4 v1.12.3
Source: [安装目录]/ Firmware

→

② Select firmware
Rerun the one-click installer: "PX4固件版本" (firmware version)
Free: 1.12, 1.14;
Full also: 1.13, 1.15–1.17

→

③ Check the version
QGroundControl (QGC): MAVLink Console
Run ver all;
read the FW version line

Packages: installer folder 2.FirmwareZip; follow each example's Readme.
For other versions, check each interface on the "Version Comparison" page.

5.5.1 PX4 Autopilot and Code Architecture: Version Comparison

Check These Five Interface Differences When Changing Versions

Page to compare	v1.12.3 (bundled default)	v1.14 and later
Multicopter Control Chain in v1.12.3	actuator_controls_0 → mixer	Torque/thrust setpoints → control_allocator
Input Levels and Directions	x/y/r∈[-1,1] Throttle z∈[0,1]	roll/pitch/yaw throttle∈[-1,1]
One-Step Output Calculation	Thrust fraction $T = z$	Thrust fraction $T = (\text{throttle} + 1)/2$
Logging, Parameters, and Communication	No uXRCE-DDS	uXRCE-DDS available (bridges to ROS 2)
Rate Controller Output Contract	actuator_controls_0 control[3]∈[0,1]	vehicle_torque_setpoint vehicle_thrust_setpoint

Readme recommendations: 0.ApiExps 3, 6, 11 use the 1.14 series; 16, 17 and 1.BasicExps/e9 use 1.12.3; 21 uses 1.13.

Chapter Outline

PX4 Architecture and Control Development Interfaces

Chapter 5
Filtering, State Estimation,
and Low-Level Control

5.1 Filtering and Low-Level Control

5.2 Sensor Calibration and Estimation

5.3 Controller Design and Allocation

5.4 Control Performance and Tuning

5.5 PX4 Architecture and Interfaces

5.6 Control Development and Testing

5.7 Other Vehicles and Advanced Cases

5.8 Exercises and Classroom Discussion

4 Units in This Section

5.5.1 PX4 Autopilot and Code Architecture

Textbook: Secs. 5.2.1.1–5.2.1.2

5.5.2 Common Interfaces and Custom Development

Textbook: Sec. 5.2.3

5.5.3 Generated Controller Integration with PX4

Textbook: Secs. 5.2.1.3–5.2.1.4

5.5.4 Interface Validation and Module Replacement

Textbook: Secs. 5.2.3, 5.3.1

5.5 PX4 Architecture and Interfaces

5.5.2 Common Interfaces and Custom Development: Four Development Interfaces

Four Interfaces: Modeling, Commands, Communication, and Advanced Modules

SimulinkPSP Toolbox
Generates PX4 code: sensor/actuator I/O, uORB read/write (6.uORB-Read-Write)
Here: "Publish and Subscribe";
"Generation and Execution"

MATLAB Command-Line Interface
Build/upload; register modules; set controllers
PX4Upload / PX4Build / PX4AppName
Here: "Controlled Message Publication" (upload)
API.pdf: "MATLAB命令行接口"

External Communication Interfaces
uORB: internal; MAVLink: ground stations/programs
"Logging, Parameters, and Communication"
8.Mavlink-Msg-Echo

RflySimPSP Advanced Library
Remote input: 15.InputSourceAPI
Actuators: 16.CtrlsSingalsAPI
State: 11.StateDataGatAPI
Here: input levels/directions + next two pages

API (application programming interface): [安装目录]/RflySimAPIs/5.RflySimFlyCtrl/API.pdf;
examples: 0.ApiExps.

5.5.2 Common Interfaces and Custom Development: Publish and Subscribe

uORB Carries Messages; Controllers Check Their Validity

Estimator: publishes attitude → **vehicle_attitude**
q[4]: attitude quaternion
timestamp: publication time
timestamp_sample: sample time
→ **Controller: subscribes independently**
→ **Logger: subscribes independently**

uORB is asynchronous publish/subscribe within the flight controller; timestamps use μs since boot. Read success ≠ valid state: check updates, instance, finite values, normalization and estimator health.

$$t_{\text{age}} = t_{\text{now}} - t_{\text{sample}} = (1006000 - 1000000)\ \mu\text{s} = 6.0\ \text{ms}$$

uORB mechanism and examples (PX4 v1.12)
https://docs.px4.io/v1.12/en/middleware/uorb.html

5.5.2 Common Interfaces and Custom Development: State and Sensor Fields

Measurements, Estimates, and Ground Truth Are Different

Data layer	Fields and units	Frame/source
IMU/combined measurements	sensor_combined.gyro_rad: rad/s accelerometer_m_s2: m/s²	Body FRD; interval averages
Body angular velocity	vehicle_angular_velocity.xyz： rad/s	Body FRD; bias corrected
Estimated attitude	vehicle_attitude.q[4]: dimensionless	Quaternion: body FRD → earth NED
Simulation ground truth	Dedicated groundtruth topics/model interfaces	Available only from specific simulation sources

IMU=inertial measurement unit; FRD=forward/right/down; NED=north/east/down. State acquisition: 0.ApiExps/11.StateDataGatAPI/Readme.pdf (recommended: 1.14 series).

5.5.2 Common Interfaces and Custom Development: Input Levels and Directions

Calibrate Remote-Control Pulse Widths, Then Map Them to Setpoints

Teaching calibration: 1100 / 1500 / 1900 μs; current value: 1700 μs.
r_{norm} is normalized stick input; ϕ_{sp} is roll setpoint. Here, maximum roll is 20° with a linear mapping.

$$r_{\text{norm}} = \frac{1700 - 1500}{1900 - 1500} = 0.5$$

$$\phi_{\text{sp}} = 0.5 \times 20^\circ = 10^\circ = 0.1745 \text{ rad}$$

v1.12.3: manual_control_setpoint uses x/y/r∈[-1,1] and throttle z∈[0,1].
Pushing x forward is positive and commands negative pitch; v1.14 and later use roll/pitch/yaw/throttle.

Remote control: 0.ApiExps/15.InputSourceAPI/Readme.pdf
External setpoints (Offboard): 0.ApiExps/17.OffboardCtrlsAPI/Readme.pdf

5.5.2 Common Interfaces and Custom Development: Actuator Control API

An Output Value Must Identify Its Interface Level

Model file in this folder	Input meaning	Downstream path
Exp1_HIL16CtrlsPWM.slx	16 PWM values (pulse-width modulation) 1000–2000	actuator_outputs_rfly →CopterSim inPWMs
Exp2_HIL16CtrlsNorm.slx	16 normalized values [-1,1]; interpreted by vehicle type	Wrapper conversion → simulated actuators
Exp3_TorqueThrustCtrls4D.slx	Three-axis torque and scalar thrust All normalized	actuator_controls_0 → mixer → actuators

Entry: 0.ApiExps/16.CtrlsSingalsAPI (recommended: 1.12.3); see “Actuator Control Example.”
A 16-value simulation vector ≠ 16 physical outputs; the hardware determines the physical output count.

5.5.2 Common Interfaces and Custom Development: One-Step Output Calculation

The Thrust Sign Depends on the Wrapper and Frame Convention

TorqueThrustCtrls: positive normalized thrust is upward; PX4 body FRD has z downward.
Here: manual rate mode (attitude loop off); T is collective thrust fraction $\in [0,1]$.
Throttle field: v1.12.3 $z \in [0,1]$; v1.14 and later throttle $\in [-1,1]$.

$$\text{v1.12.3}: \ T = z = 0.5; \qquad \text{v1.14 and later:} T = \frac{\text{throttle} + 1}{2} = \frac{0+1}{2} = 0.5$$

$$\mathbf{thrust}_{\text{body}} = [0,\ 0,\ -T] = [0,\ 0,\ -0.5], \qquad \text{control}[3] = T = +0.5$$

actuator_controls_0 → **Mix by airframe** → **Individual motor values and hardware outputs**

0.5 is a normalized thrust fraction, not N; mixing converts it to individual motor values.

5.5.2 Common Interfaces and Custom Development: Actuator Control Example

Use CH5 to Verify That the Generated Module Takes Control

Entry: 0.ApiExps/16.CtrlsSingalsAPI/Readme.pdf (recommended: 1.12.3)
Hardware-in-the-loop (HITL; HIL in the textbook)

① Build/upload	② Start HITL	③ PX4 control	④ Simulink control
Build Exp1 (PWM); use PX4Upload	Set CH5 lowest; COM port in HITLRun	QGC takeoff; remote control works	Disarm; CH5 highest Auto-arm, then take off

Expected behavior and checks

CH5 highest: QGC takeoff inactive; throttle climbs
CH5 lowest: Loiter hover; QGC landing works; PX4 regains authority
Repeat Exp2/Exp3: same behavior

Failures and checks

Build: diagnostic window; MATLAB≥2022b
Upload: USB, COM port, build command
CH5 ineffective: channel setup, isEnCtrl
Caution: highest CH5 automatically arms

Check: CH5 switches authority between PX4 and Simulink; only one is effective at a time.

5.5.2 Common Interfaces and Custom Development: Logging, Parameters, and Communication

Logs Record Evidence; Parameters Change Runtime Values

Purpose	Entry or mechanism	What to check
Save data	0.ApiExps/5.Log-Write-Read px4demo_log.slx: writes .bin px4demo_log_qgc.slx: writes .ulg	Setpoints, states, requested/actual inputs Use one time base.
Change parameters	0.ApiExps/10.QGC-Param-Tune PX4QGCTune.slx	Defaults: FLT=100, INT=50. Check received values and when changes take effect.
Internal/external communication	uORB: internal flight-controller topics MAVLink: ground stations and external programs	Data direction, topic and sample age Network delay and packet loss

SL_RFLY_FLT/INT: example-defined float/integer parameters; edit in QGroundControl.
uXRCE-DDS (v1.14 and later): flight-controller client + PC agent bridges selected uORB topics.

Chapter Outline

PX4 Architecture and Control Development Interfaces

Chapter 5
Filtering, State Estimation,
and Low-Level Control

5.1 Filtering and Low-Level Control

5.2 Sensor Calibration and Estimation

5.3 Controller Design and Allocation

5.4 Control Performance and Tuning

5.5 PX4 Architecture and Interfaces

5.6 Control Development and Testing

5.7 Other Vehicles and Advanced Cases

5.8 Exercises and Classroom Discussion

4 Units in This Section

5.5.1 PX4 Autopilot and Code Architecture

Textbook: Secs. 5.2.1.1–5.2.1.2

5.5.2 Common Interfaces and Custom Development

Textbook: Sec. 5.2.3

5.5.3 Generated Controller Integration with PX4

Textbook: Secs. 5.2.1.3–5.2.1.4

5.5.4 Interface Validation and Module Replacement

Textbook: Secs. 5.2.3, 5.3.1

5.5.3 Generated Controller Integration with PX4: Model and Firmware Relationship

Generated Code Joins the PX4 Runtime as a Module

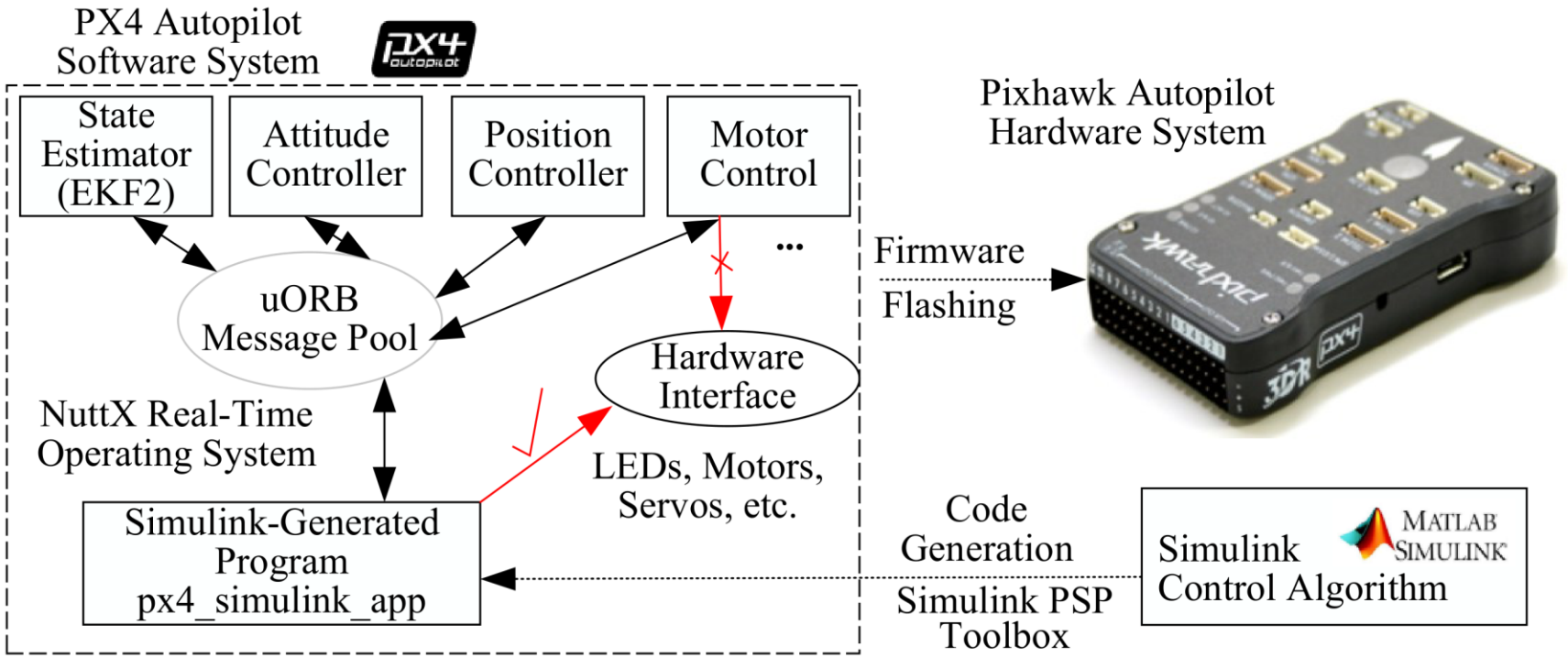
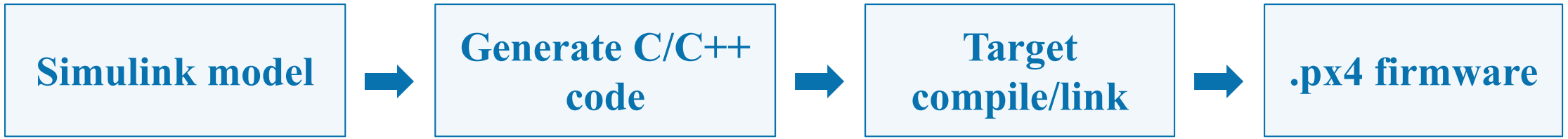


Fig. 5.6 Relationship between Simulink Controller and PX4 Firmware

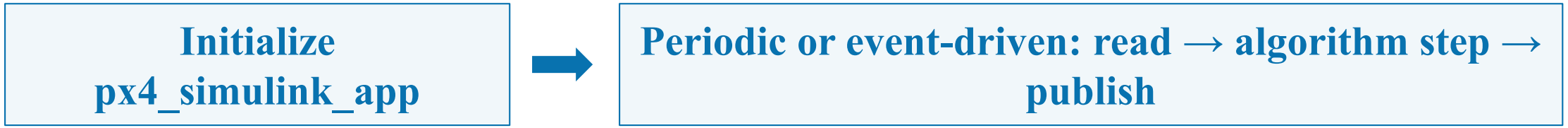
Red ✗: native motor control output is masked; red ✓: px4_simulink_app takes over hardware output.

5.5.3 Generated Controller Integration with PX4: Generation and Execution

Modeling, Code Generation, Firmware, and Execution Are Separate Stages



TLC: Target Language Compiler; provides code-generation templates.
Top: generation; bottom: execution. Generated code is compiled into firmware, then scheduled.



Check three kinds of evidence separately: code-generation report, built .px4 firmware and signals on the board.

Entry: 0.ApiExps/3.DesignExps/Readme.pdf (recommended: 1.14 series)
Main model: Exp4_AttitudeSystemCodeGen.slx.

5.5.3 Generated Controller Integration with PX4: Control Period and Time Budget

Measure Step Size, Execution Time, and Data Age Separately

Teaching timing (classroom values): nominal period 4 ms, nominal frequency 250 Hz.

Stage	Time/ms	Operation
Read	0.10	Get the latest valid sample
Compute	0.65	One controller step
Publish	0.15	Set sample and publication timestamps
Total/margin	0.90 / 3.10	0.10+0.65+0.15=0.90； 4.00-0.90=3.10

$$t_{\text{age}} = t_{\text{now}} - t_{\text{sample}} = (1008600 - 1008000)\,\mu\text{s} = 0.6\,\text{ms}$$

Intervals 4.1/3.9 ms: jitter ±0.1 ms; count worst-case execution time and missed deadlines.

5.5.3 Generated Controller Integration with PX4: Control Rate and State Freshness

Control Rate Sets How Often We Compute, Not How Fresh the State Is

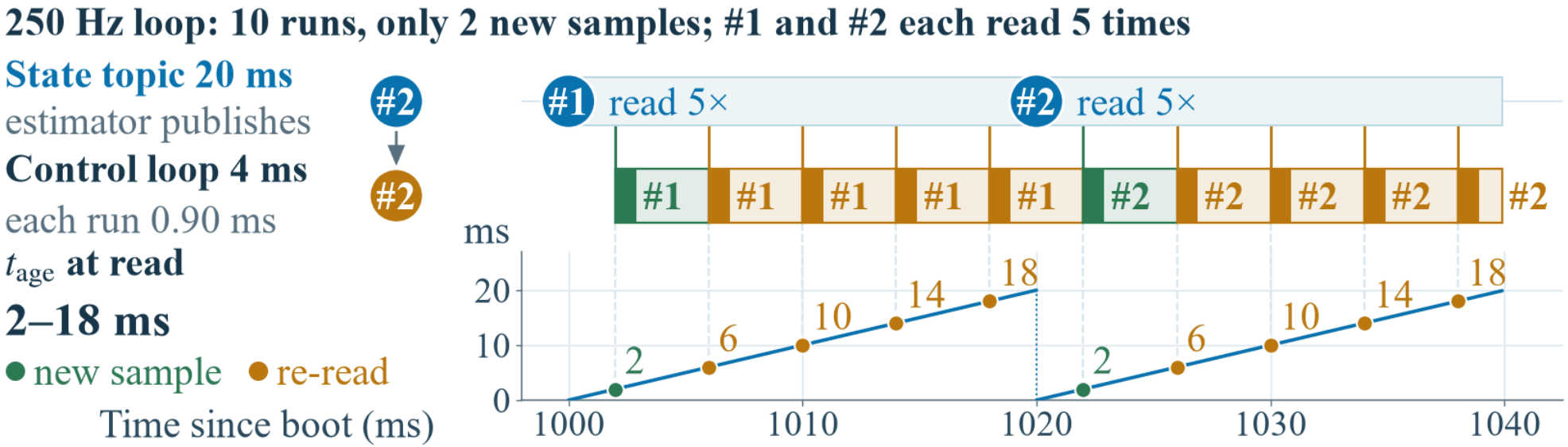


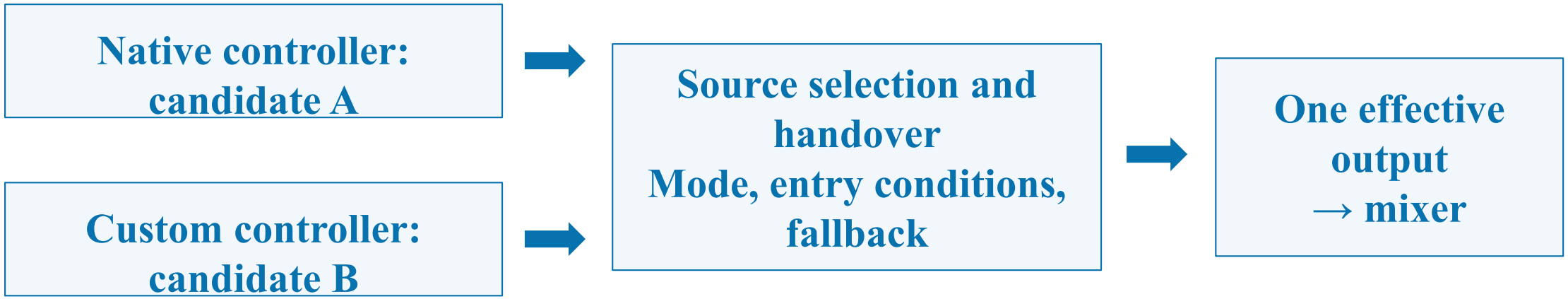
Figure: Control rate and state freshness animation (schematic; sample-to-publication delay = 0)

Sample IDs: each is read 5 times before a new state arrives; only 2 of 10 reads get new samples.

Age at each read: repeated reads add 4 ms, rising from 2 ms to 18 ms before resetting.

5.5.3 Generated Controller Integration with PX4: Output Authority and Arbitration

Parallel Controllers Need One Effective Control Source



Standard uORB only carries messages: both paths may compute, but only one may write the effective output.

Installer item 10: “是否屏蔽PX4自身控制器输出” (mask native PX4 output) → “是” (Yes).
PSP (PX4 hardware support package): see “Native Output Masking Options” for choices and four functions.

5.5.3 Generated Controller Integration with PX4: Native Output Masking Options

The Installer's Masking Option Determines Who Drives the Motors

SimulinkPSP: customization, code generation, safe control switching, cross-platform compatibility
Installer item 10: "是否屏蔽PX4自身控制器输出" (mask native PX4 output)
Textbook: "mask the PX4 motor-control code," last item in Fig. 2.4.

"是" (Yes): masks Firmware motor-control signals to prevent conflict with generated code.

"否" (No): keeps native outputs for PX4's own algorithms and official firmware.
"自动" (Auto): updates only changed files.
actuator_controls_0: masks only this uORB message; supports FMU, DShot and different vehicle types.
"自定义屏蔽" (Custom mask): an xlsx file replaces selected files/strings to mask a module's output.

Software-in-the-loop (SITL; SIL in the textbook): PX4_SITL controller output stays active.
Restore official firmware: rerun the installer with "否" (No), or run PX4Official, then PX4Upload.

5.5.3 Generated Controller Integration with PX4: Full and Partial Replacement

The Interface Level Defines Which Control Responsibilities Remain

Integration level	Custom module handles	Usually retained/to check
External position setpoints	Reference trajectory and mission logic	Native position, attitude, rate control and mixing
Partial rate-control replacement	Rate error → torque Retain upstream thrust setpoint	Estimation, upstream setpoints, mixing and safety
Full position/ attitude control	Control loops from position to torque/thrust	Estimation/output interfaces; full mode handover

Offboard: external-computer setpoints; “External Setpoint Example,” 0.ApiExps/17.OffboardCtrlsAPI.

A clear replacement boundary helps identify which algorithm caused the result.

5.5.3 Generated Controller Integration with PX4: External Setpoint Example

Offboard Changes the Setpoint Source; Native Control Remains

Entry: 0.ApiExps/17.OffboardCtrlsAPI/Readme.pdf (recommended firmware: 1.12.3)

① Build/flash
Exp1 position model;
flash flight controller

→

② Start HITL
CH5 lowest;
HITLRun as administrator

→

③ Initialize EKF
Enter COM port;
wait for EKF to finish

→

④ Enter Offboard
CH5 highest;
auto-arm after 1 s

Expected behavior (Exp1 position setpoints)
Roll stick fully right: 10 m right
Pitch stick fully forward: 10 m forward
Throttle fully up: climb to 20 m altitude
Yaw stick fully right: yaw π rad

Failures and checks
Missing build library: toolchain and MATLAB
HITL: administrator rights; firewall
No Offboard: CH5 and firmware version
Caution: highest CH5 automatically arms

EKF = extended Kalman filter. Check: the aircraft holds its preset position in QGC; native PX4 modules still perform downstream control and mixing.

Chapter Outline

PX4 Architecture and Control Development Interfaces

Chapter 5
Filtering, State Estimation,
and Low-Level Control

5.1 Filtering and Low-Level Control

5.2 Sensor Calibration and Estimation

5.3 Controller Design and Allocation

5.4 Control Performance and Tuning

5.5 PX4 Architecture and Interfaces

5.6 Control Development and Testing

5.7 Other Vehicles and Advanced Cases

5.8 Exercises and Classroom Discussion

4 Units in This Section

5.5.1 PX4 Autopilot and Code Architecture

Textbook: Secs. 5.2.1.1–5.2.1.2

5.5.2 Common Interfaces and Custom Development

Textbook: Sec. 5.2.3

5.5.3 Generated Controller Integration with PX4

Textbook: Secs. 5.2.1.3–5.2.1.4

5.5.4 Interface Validation and Module Replacement

Textbook: Secs. 5.2.3, 5.3.1

5.5.4 Interface Validation and Module Replacement: Minimal Input Validation

Read-Only Checks Start with Direction, Units, and Time

Pure yaw here: roll = pitch = 0; quaternion order w/x/y/z.
In general, body angular velocity differs from Euler angle rates.

$$\mathbf{q} = [0.9961947,\ 0,\ 0,\ 0.0871557], \quad \omega_z = 0.2\ \text{rad/s}$$

$$\|\mathbf{q}\|_2 \approx 1, \quad \psi = 2\,\text{atan2}(0.0871557,\ 0.9961947) = 10^\circ$$

Check	Expected interpretation
Direction and units	FRD→NED attitude; ω_z=0.2 rad/s
Sample age	1006000-1000000=6000 μs, or 6 ms

Read: 0.ApiExps/11.StateDataGatAPI/Readme.pdf; publishing: “Controlled Message Publication.”
SITL console (read-only): listener vehicle_attitude -n 5; uorb top.

5.5.4 Interface Validation and Module Replacement: Controlled Message Publication

After Publishing, Verify Received Values and Shutdown Status

Isolated HITL: disarmed; no actuators attached. Exp3_uORB_write.slx (recommended: 1.14 series).

① **Build** Exp3 model → ② **USB upload** PX4Upload → ③ **HITLRun** Await EKF init. → ④ **QGC** MAVLink Console

Expected check	Disarmed branch/action	Receiver check
control[0] / [3] / [7]	1 / 4 / 8	listener rfly_px4 -n 5 float32[8]; check array indices
timestamp / update rate	uint64, μs since boot Rate follows model schedule	uorb top -a; Enter exits Time increases; measure rate
Normal shutdown	px4_simulink_app stop px4_simulink_app status	not started; verify rate = 0 Reject stale retained samples by age

Failures: missing command or rate after stop → stay disarmed; check uploaded firmware/publishers.

Entry: 0.ApiExps/6.uORB-Read-Write/Readme.pdf, Sec. 5.3 (pp. 9–13).

5.5.4 Interface Validation and Module Replacement: Rate Controller Input Contract

Input Contracts Specify Fields, Units, and Failure Conditions

Interface contract: agreed inputs/outputs, units, frames, rates and timing between modules.

Input/publisher	Fields and units/frame	Validity and handling
Angular velocity **Sensor processing module**	vehicle_angular_velocity.xyz rad/s, body FRD	Finite fields; sample age≤8 ms. Use timestamp_sample.
Rate/thrust setpoints Selected upstream source Body FRD	vehicle_rates_setpoint roll/pitch/yaw: rad/s thrust_body (3D): normalized	Finite fields; each thrust axis [-1,1]. Setpoint age≤100 ms. Use timestamp.
Control mode commander	vehicle_control_mode flag_armed flag_control_rates_enabled	Armed; rate loop on; module selected Else: stop publishing; fall back.

Here: v1.12.3 teaching rate controller; non-manual branch; period 4 ms.
Teaching thresholds: 8 ms (sample age), 100 ms (setpoint age).

5.5.4 Interface Validation and Module Replacement: Rate Controller Output Contract

Output Contracts Define Both Values and Publication Authority

Output/consumer	Meaning and teaching limits	Publication rules
actuator_controls_0 control[0]–[2]→mixer	Normalized roll/pitch/yaw torque Here, each axis [-1,1]; not N·m	Selected control source only; timestamp_sample records gyro sample time.
actuator_controls_0 control[3]→mixer	Normalized throttle (collective thrust) Range [0,1]; positive upward	Pass through upstream thrust; torque/thrust: same control step; publication time: timestamp

$$\text{control}[0]\sim[3] = [0.10,\ -0.05,\ 0,\ 0.50]$$

$$\Delta t = t_{\text{pub}} - t_{\text{sample}} = (1008900 - 1008000)\ \mu\text{s} = 0.9\ \text{ms}$$

No publisher field in actuator_controls_0: verify source via diagnostics, mode configuration and publication paths.

5.5.4 Interface Validation and Module Replacement: Control Source Switching Calculation

Align Controller States Before Switching to Reduce Output Jumps

One axis, normalized torque: native 0.24; custom proportional term 0.10, integral contribution 0.04. u_I is the integral term's output contribution, not the integrated error state.

$$u_{\text{custom}} = 0.10 + 0.04 = 0.14, \quad \Delta u = 0.14 - 0.24 = -0.10$$

$$u_{I,\text{aligned}} = 0.24 - 0.10 = 0.14 \quad \Rightarrow \quad u_{\text{custom}} = 0.10 + 0.14 = 0.24$$

Handover stage	Quantities that must match or be defined
Entry	Setpoints, state age, integral/filter states, output limits
Switching/failure	One control source; handover time; fault exit and traceable fallback

See "Control Source Switching Example": 0.ApiExps/21.SwitchPX4Simulink (firmware 1.13).

5.5.4 Interface Validation and Module Replacement: Control Handover Sequence

Align the Integral Contribution, Then Switch the Output Source

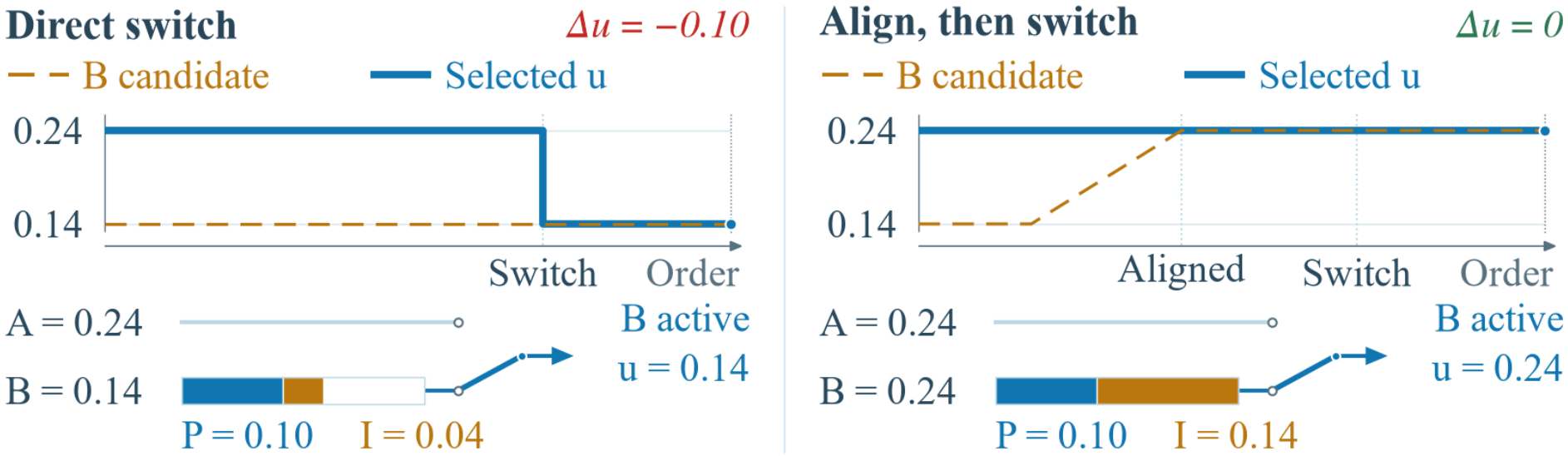


Figure: Direct handover versus handover after integral prealignment (single-axis PI schematic)

Sequence: B adjusts its integral contribution from 0.04 to 0.14; A still outputs 0.24 during alignment.
Takeover: select B only when both candidates match; dashed lines show candidates, not simultaneous actuation.

5.5.4 Interface Validation and Module Replacement: Control Source Switching Example

RePX4Block Uses a Remote-Control Channel to Select PX4 or Simulink

Entry: 0.ApiExps/21.SwitchPX4Simulink/Readme.pdf (firmware 1.13)

TorqueThrustCtrls outer isEnCtrl is Boolean 0/1: CH8<1500 gives 0; CH8≥1500 gives 1.
With auto masking enabled, internal RePX4Block mode is 0/2, selecting PX4/Simulink respectively.

① Build	② Rebuild	③ Upload/start	④ Verify switch
Init_control.m Build Exp6	Run Firmware/ 6x_Start.bat	PX4Upload; COM port in HITLRun	QGC: arm, take off; move CH8

Expected behavior and checks
QGC on switch to Simulink:
Switching to Simulink control!
Switch back: Switching to PX4 control!
CH1–CH4 work with either controller

Failures and checks
No message: 6x_Start.bat; upload again
No switch: isEnCtrl, mode, CH8
Auto masking on; auto Loiter and arm off
QGC airframe: HIL Quadcopter X

Extension: check jumps with integral prealignment (“Control Source Switching Calculation”).

5.5.4 Interface Validation and Module Replacement: Partial Replacement Acceptance Checks

Replacement Needs Evidence from Modules, Messages, and the Closed Loop

① Module boundary	② Publication authority	③ Signals and timing	④ Modes and fallback	⑤ Closed-loop results
Set version and loop; keep estimation, mixing and the safety chain	Native path exits as designed; only custom output active.	Fields, frames, units, rates, data age: meet contract.	Entry and stop, failure handling, traceable fallback	Same baseline: compare response, inputs, limits, stability margins.

Pass all five levels in order; investigate any failure before tuning control parameters.

Version: v1.12.3 replaces publication of actuator_controls_0; the replacement point changes in v1.14 and later.

See “Module Replacement Example”: 1.BasicExps/e9-ReplacePX4Module (recommended: 1.12.3).

5.5.4 Interface Validation and Module Replacement: Module Replacement Example

After Masking Native Output, Simulink Publishes actuator_controls_0

Entry: 1.BasicExps/e9-ReplacePX4Module/Readme.pdf (firmware 1.12.3)

① **Prepare:** follow Readme’ s one-click installer steps; select firmware 1.12.3.

② **Mask:** comment out both publish calls in MulticopterRateControl.cpp; mark variables UNUSED.

③ **Flash:** build for the board in Windows Subsystem for Linux (WSL); flash with QGC. Calibrate the radio and set flight modes.

④ **Confirm:** after a valid HITL baseline, verify native publication stops. Failure to take off alone does not prove masking.

⑤ **Integrate:** run Init_control.m, build Exp6, upload with PX4Upload.

⑥ **Fly:** radio in Stabilized mode (CH5 or CH6); arm and take off.

Expected: takeoff and attitude control work, usually worse than native control.
Subscribes: angular velocity, attitude, remote input; publishes actuator_controls_0.

Build fails: check UNUSED. No takeoff: Stabilized mode.
Recovery: restore source or rerun the one-click installer.

Classroom Discussion

Output Alternates Between Two Value Sets: What Should You Check First?

Order	Suggested checks
1 Establish a baseline	Stop uncontrolled motion; return to a known safe simulation condition
2 Assign responsibility	Fix the version, mode, output instance and selected control source
3 Trace publication	Check module status and uorb top rates; timestamps cannot identify publishers
4 Verify handover	Keep one path; check whether jumps stop, then test switching and fault fallback
5 Validate the closed loop	Restore one control source and repeat the same condition; the jumps should disappear

Confirm control authority before judging the control law: changing messages only show that someone is publishing.

Section Summary

Flight Controller Integration Needs Clear Data and Control Authority

Read data correctly → **Publications received correctly** → **Correct closed loop and modes**

Architecture: sensing→estimation→control→mixing/allocation→output; uORB inside, MAVLink outside.

Input contract: source, fields, units, frames, data age and failure handling.

Output contract: meaning of normalized values, downstream allocation, one publisher (mask native output) and fallback.

Versions: bundled default v1.12.3 uses mixers; v1.14 and later use a control allocator.

Common errors: quaternions treated as Euler angles; normalized values as SI quantities; control rate as state freshness; successful publication as correct closed-loop behavior.