FFair_Link

User Manual

Desktop application v0.8.1 • Arduino library v0.9.0

From installation to a working MSFS or X-Plane hardware controller

For users who are comfortable with simulator configuration, Arduino sketches, basic electronics and structured profiles, without requiring firmware-level expertise.

English edition • August 2026

Contents

Select a numbered chapter title in Word to jump directly to it.

About this manual

This manual describes the complete user workflow for FFair_Link v0.8.1 and the bundled FFairLink Arduino library v0.9.0. It covers Microsoft Flight Simulator through FSUIPC7, X-Plane 12 through its local Web API, Serial COM and UDP devices, profile construction, expressions, rules, commissioning and fault finding.

Scope: The desktop application and Arduino library have independent version numbers. This manual states both. Do not assume that a profile, sketch or firmware from a different release exposes exactly the same fields or protocol capabilities.

Intended reader

The intended reader can install simulator utilities, identify a Windows COM port, upload an Arduino sketch and perform basic low-voltage wiring. Firmware internals, C# development and advanced network administration are outside the normal workflow, although the reference sections expose enough detail to diagnose most installations.

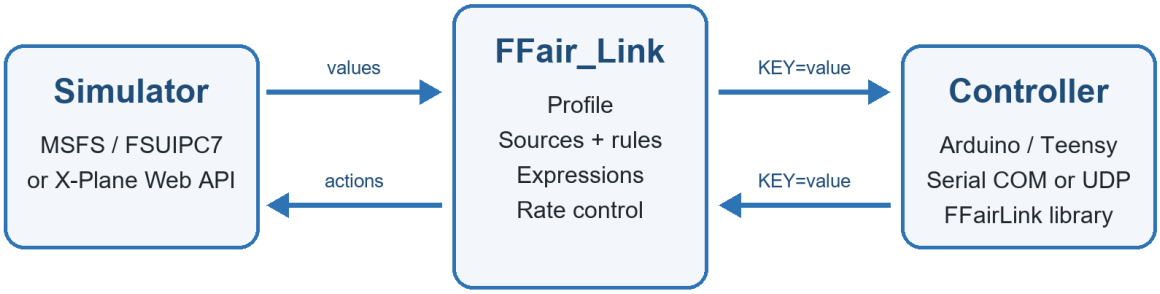
Safety and backups

- Disconnect power before changing wiring. Observe the voltage and current limits of the controller, connected modules and power supply.
- Use correctly rated external supplies for connected hardware when required; join grounds only as specified by the hardware design.
- Clone a working profile before experimental changes and keep a copy of the complete profiles folder.
- Keep wiring and connected hardware in a safe, observable state during commissioning.

Terminology

Term	Meaning
Profile	A simulator-specific collection of one or more device definitions.
Device	One physical controller and its transport, identity and variables.
Variable / item	One data path between the simulator and a device.
Source	The simulator value read by a Sim → Device item.
Action	The simulator operation performed by a Device → Sim rule.
Key	The short protocol name used on the wire, for example BUS or PB.

How FFair_Link works



FFair_Link data flow

The controller does not need to understand FSUIPC offsets, LVars, X-Plane datarefs or commands. It exchanges compact messages such as BUS=1 or PB=0. The active FFair_Link profile translates between those stable device keys and the current simulator or aircraft interface.

Direction	Processing chain	Typical example
Sim → Device	Read source → test condition → evaluate expression → format payload → rate-limit → transmit	Electrical bus state becomes BUS=1 and drives an LED.
Device → Sim	Receive key/value → evaluate input transform → evaluate ordered rules → execute one or more actions	A switch sends PB=1 and a parking-brake action is executed.

Why profiles are useful

A sketch can remain unchanged while different profiles map the same keys to different aircraft. A key such as PB is a device contract; the MSFS profile can map it to an FSUIPC control while an X-Plane profile maps it to a writable dataref. Reuse is reliable only when the JSON profile and the sketch agree on device ID, transport and keys.

Install FFair_Link

System requirements

- Windows 10 or Windows 11, 64-bit.

- The self-contained FFair_Link v0.8.1 Windows x64 distribution; no separate .NET installation is required.
- For MSFS: a working FSUIPC7 installation. LVar use also requires the FSUIPC WASM/WAPI components and the distributed FSUIPC_WAPID.dll beside FFair_Link.exe.
- For X-Plane: X-Plane 12 with its local Web API reachable at localhost:8086.
- For firmware upload: Arduino IDE and the board core/USB driver required by the selected Arduino-compatible board.

Application installation

1. Download the current FFair_Link distribution ZIP.
2. Extract the entire ZIP to a normal writable folder. Do not run FFair_Link.exe from inside the ZIP.
3. For beta use, avoid a protected location such as Program Files; a dedicated folder such as C:\FFair_Link is easier to back up and update.
4. Keep the folder structure intact. Profiles, examples, the Arduino library ZIP and native dependencies are distributed alongside the executable.
5. Start FFair_Link.exe. If Windows displays a security prompt, verify the file came from the intended release before allowing it.

Default startup: A clean v0.8.1 installation starts with the MSFS Tutorial Minimum Cessna 152 profile, simulator and device auto-connect disabled, and blank COM fields. This prevents accidental connection to the wrong board.

Updating

Extract a new release to a separate folder first. Copy or import only the profiles you intend to retain, then test the minimum example before retiring the previous installation. Keeping the previous folder makes rollback immediate and preserves known-good JSON files.

First working panel: switch and LED

The bundled minimum examples deliberately contain one output and one input. Complete this test before building a large panel; it validates the simulator link, device link, handshake, keys, formulas and wiring with very little ambiguity.

Hardware

- One supported Arduino-compatible board.

- A USB cable for the Serial example, or a supported Ethernet interface for UDP.
- One switch connected between digital pin 2 and GND. The sketch uses INPUT_PULLUP.
- The board's built-in LED, used as the output indicator.

MSFS minimum Serial test

1. Install the FFairLink library ZIP and open the bundled MINIMUM01 Serial example.
2. Select the correct board and port in Arduino IDE, compile and upload the sketch.
3. Start MSFS, load the default Cessna 152, and start FSUIPC7.
4. In FFair_Link select MSFS and Tutorial Minimum Cessna 152.
5. Open device MINIMUM01, select its COM port, confirm Device ID MINIMUM01 and save.
6. Connect FSUIPC, then connect devices. The order is not critical, but this sequence makes the log easier to read.
7. Change aircraft electrical power. The built-in LED follows the BUS value.
8. Operate the pin-2 switch. PB=0 or PB=1 is sent and the parking brake changes.

X-Plane minimum tests

Use Tutorial Minimum Cessna with device MINIMUM01 for Serial, or MINIMUM01_UDP for UDP. The example reads sim/cockpit2/electrical/battery_on and writes sim/cockpit2/controls/parking_brake_ratio. The Serial and UDP JSON files use keys that match their corresponding sketches.

Expected identity: MINIMUM01 and MINIMUM01_UDP are different device IDs. A UDP sketch compiled as MINIMUM01_UDP will not satisfy a profile expecting MINIMUM01, even when the data keys are identical.

Main window and operating sequence

Control	Purpose
Simulator	Select MSFS or X-Plane 12. This changes the available profile family and item types.
Profile	Select the aircraft/hardware mapping. New, Edit, Clone and Delete manage profiles.
Auto-connect simulator	Connects the selected simulator automatically when appropriate.
Auto-connect devices	Attempts to connect the devices in the active profile automatically.
Connect FSUIPC / X-Plane	Starts or stops the simulator-side connection.

Control	Purpose
New / Find / Reload devices	Creates a device, discovers FFairLink identities on COM ports, or reloads profile files.
Log	Primary evidence for connection, handshake, CRC, profile and simulator errors.

Recommended operating sequence

1. Start the simulator and load the intended aircraft.
2. Start FSUIPC7 for MSFS, or confirm X-Plane's local Web API is available.
3. Start FFair_Link and select the correct simulator and profile.
4. Connect the simulator interface and check its status.
5. Power/connect the controllers, then connect devices or use Find.
6. Confirm each device reports Connected before operating the panel.
7. Keep the log visible during initial commissioning or after profile changes.

Connection order: The application and library tolerate either connection order. Retained Arduino outputs are sent after a new handshake, so connecting the controller before or after the simulator is normally safe.

Profiles

A profile belongs to one simulator family and contains one or more device JSON files. Use a separate profile whenever an aircraft exposes different offsets, LVars, datarefs or commands, even if the physical panel is unchanged.

Action	Use
New	Create an empty profile for the selected simulator.
Edit	Change profile display information and auto-connect options.
Clone	Create a complete working copy before modifications or as a starting point for a similar aircraft.
Cross-simulator clone	Copies common structure but clears or converts simulator-specific targets. Every item must be reassigned and tested.
Delete	Removes the profile. Back up the profile folder first if it may be needed later.

Profile layout

```
profiles/
  msfs/
    My Aircraft/
      profile.json
      devices/
        PANEL01.json
  xplane/
    My Aircraft/
      profile.json
      devices/
        PANEL01.json
```

The normal workflow is to edit profiles through FFair_Link. Direct JSON editing is useful for review, version control and bulk inspection, but a malformed file can prevent a profile or device from loading.

Included profiles

Simulator	Profile	Purpose
MSFS	Tutorial Minimum Cessna 152	Small Serial switch/LED validation profile.
X-Plane	Tutorial Minimum Cessna	Serial and UDP switch/LED examples.

Profile discipline

- Keep device IDs stable; rename only when the sketch/firmware is updated at the same time.
- Give items short, descriptive IDs and unique protocol keys.
- Clone before large rule or formula changes.
- Test one direction at a time: Sim → Device first, then Device → Sim.
- Record the target aircraft version when an add-on uses private LVars or datarefs.

Devices, Serial COM and UDP

Field	Meaning / recommendation
Device ID	Must match the ID compiled into the sketch or firmware. Comparison is case-insensitive, but identical spelling is recommended.
Version	Expected/documented firmware version. The handshake can report the running version.
Transport	Serial COM or UDP.
COM port	Windows port for Serial. Leave blank in distributable generic profiles.

Field	Meaning / recommendation
UDP IP	Controller IP. Leave blank to use broadcast discovery.
UDP port	Local controller port; bundled examples use 49000.
Validate handshake	Checks FFAIRLINK:deviceId before treating the connection as valid.
Poll rate	Supported values are 50, 25, 10, 5, 2 or 1 Hz; 10 Hz is a practical starting point.
DTR	Normally off. Some boards reset when DTR is asserted; discovery may retry an unresponsive port with DTR.
Send CRC8	Recommended. Adds *XX and detects corrupted messages.

Serial discovery

Find scans present, unreserved COM ports at 115200 baud. It first attempts identification without resetting each controller, then retries an individual unresponsive port with DTR when needed. A valid identity is assigned only to a matching device in the active profile.

Handshake

```
Request:  FFAIRLINK:PANEL01
Identity: FFAIRLINK:PANEL01;VER=1.0;...
```

Handshake validation prevents a working but wrong controller from receiving another device's traffic. Disable validation only for a deliberately simple or legacy endpoint that cannot answer FFairLink identity requests.

Transport selection

Choose Serial when…	Choose UDP when…
The controller is near the PC; simple USB cabling is preferred; low setup overhead matters.	Network cabling is more convenient; the cockpit is distributed; electrical USB limitations are undesirable.
A fixed COM assignment is acceptable or Find can identify the board.	The board has reliable Ethernet/Wi-Fi support and the local network is under your control.

Variables and data flow

Sim → Device items

Field	Purpose
ID	Unique internal item name. Other same-device values may reference it.
Type / source	Offset, LVar, dataref or derived source, depending on simulator.
Condition	Optional expression. Zero is false; a non-zero result allows processing.
Expression	Transforms the raw simulator value. The raw value is [value].
Payload	Wire template, normally KEY={value}. It may use {key}, {value} and available same-device item placeholders.
Minimum change	Suppresses insignificant changes.
Minimum interval	Limits how frequently the item can be transmitted.
Heartbeat	Forces a periodic refresh; zero disables heartbeat.

Device → Sim items

The input key selects the item. The raw wire value is [raw]. The top-level transform produces [value]. Ordered rules then evaluate conditions and execute simulator-specific actions. The editor supports up to 16 actions per item.

Ordering and dependencies

An expression can reference another item in the same physical device only when that value is already available. Keep dependency-producing items before dependent items, avoid circular references, and use the Test controls before enabling live output.

Transfer-rate tuning

Signal	Suggested starting point
Switch / discrete lamp	minChange 1; minInterval 20–100 ms; heartbeat 1000–3000 ms if periodic confirmation is useful.
Slow analog gauge	minChange matched to visible resolution; minInterval 50–250 ms; heartbeat 1000–3000 ms.

Signal	Suggested starting point
Fast-changing analog input	Small minChange; minInterval selected from measured control response; avoid unnecessary heartbeat.
Static configuration	Low poll rate and no heartbeat unless the endpoint can lose state.

Microsoft Flight Simulator

Connection

1. Start MSFS and load the aircraft.
2. Start FSUIPC7 and wait until it is connected to the simulator.
3. In FFair_Link select MSFS and the intended profile.
4. Select Connect FSUIPC and check the log for a successful connection.
5. Use Browse Offsets, Browse LVars and the control selector to identify real targets; do not guess aircraft-specific names.

MSFS Sim → Device sources

Item type	Use
offset	Read an FSUIPC offset using its address and data type.
lvar	Read a Local Variable through the FSUIPC WASM/WAPI interface.
derived	Build a value from expressions and other available items without a direct simulator source.
setLvar / setOffsetBit	Legacy readable profile forms may exist; new output and input logic should be created with the current editors and rules.

Offset data types

Type	Meaning
UB / SB	Unsigned / signed 8-bit integer.
UW / SW	Unsigned / signed 16-bit integer.
UD / SD	Unsigned / signed 32-bit integer.
DD	64-bit integer.

Type	Meaning
FLT / DBL	32-bit / 64-bit floating-point value.
STR	String. Use only when the required explicit length and profile support have been verified; numeric items are the normal path.

MSFS Device → Sim actions

Action	Target / behavior	Context value
setControl	Send an FSUIPC/MSFS control with an integer parameter. Decimal expression results are rounded away from zero.	[value]
setLvar	Write a named LVar.	[lvar] is the current target LVar value.
setOffset	Write an FSUIPC offset using its selected data type.	[offset] is the current target offset value.
setOffsetBit	Set or clear a bit in a UW offset using a valid mask.	[offset] is the current UW value.

LVar availability: If Browse LVars is empty or LVar reads fail, verify the aircraft is fully loaded, the FSUIPC WASM module is installed and FSUIPC_WAPID.dll is present beside the FFair_Link executable.

X-Plane 12

Connection

1. Start X-Plane 12 and load the aircraft.
2. Confirm the local Web API is reachable at localhost:8086 and is not blocked by security software.
3. Select X-Plane 12 and the intended profile in FFair_Link.
4. Connect X-Plane and use the Dataref and Command browsers to select actual aircraft targets.

X-Plane Sim → Device sources

Item type	Use
dataref	Read a scalar or selected dataref value through the X-Plane interface.

Item type	Use
derived	Generate a value from expressions and already available items.

X-Plane Device → Sim actions

Action	Use
setDataref	Write a writable dataref. [dataref] exposes the current value of the selected target in the rule expression.
command	Activate an X-Plane command as a pulse or timed/continuous hold.

Command timing

Hold setting	Behavior
Blank	One activation pulse.
1–60000 ms	Hold for the specified finite time.
∞, -1, inf or infinite	Hold while the rule condition is true; release when it becomes false.
Repetitions 1–99	Allowed for a finite hold. More than one repetition requires a finite hold; infinite hold forces one repetition.

Writable target: A dataref may be visible in the browser but read-only. A correct name does not guarantee that X-Plane or the aircraft permits writing it.

Rules and simulator actions

Every Device → Sim item uses a rule set. Rules run from top to bottom and contain an optional condition plus an action-specific expression and target.

Mode	Behavior	Typical use
firstMatch	Execute the first rule whose condition is true, then stop.	Mutually exclusive switch positions or prioritized behavior.

Mode	Behavior	Typical use
allMatches	Execute every rule whose condition is true.	One physical event must update several simulator targets.

Rule evaluation

1. Receive KEY=value from the device.
2. Expose the received number as [raw].
3. Evaluate the item's transform expression to produce [value]. If blank, [raw] is used.
4. Evaluate each rule condition in order. A blank condition is always true; zero is false and a non-zero numeric result is true.
5. Evaluate the rule expression. If blank, [value] is used.
6. Execute the selected action and continue or stop according to allMatches/firstMatch.

Examples

Purpose	Condition	Expression / action
Set parking brake from switch	[value] != 0	setControl; parameter 1
Clear parking brake	[value] == 0	setControl; parameter 0
Three-position selector	[value] == 2	setLvar; expression 2
Toggle a target using its current state	[value] != 0	setLvar; expression if([lvar] == 0, 1, 0)
Write one offset bit		setOffsetBit; valid UW offset and mask

Avoid repeated toggles: A continuously true rule that performs a toggle can oscillate every time the same input is processed. Prefer explicit 0/1 writes, edge-producing firmware, a pulse input, or a condition that is true only for the intended event.

Expressions and formulas

FFair_Link evaluates formulas with NCalc 6.2 plus three application functions: Bit, Bcd and Mod. Use a dot as the decimal separator. Final transform and action results should be numeric; conditions may produce booleans or numeric 0/1 results.

Parameters available in expressions

Parameter	Available in	Meaning
[value]	Sim → Device transform; rules after input transform	Current working value.
[raw]	Device → Sim item transform	Numeric value received from the wire.
[itemId]	Same physical device when already available	Latest value of another item.
[lvar]	MSFS setLvar rule	Current target LVar value.
[offset]	MSFS setOffset / setOffsetBit rule	Current target offset value.
[dataref]	X-Plane setDataref rule	Current target dataref value.

Operators

Group	Operators
Arithmetic	+ - * / % **
Comparison	= == != <> < <= > >=
Logical	and && or || not !
Bitwise	& | ^ ~ << >>
Membership / text	IN, NOT IN, LIKE, NOT LIKE (less common in numeric FFair_Link items)

Functions

Function	Purpose / example
if(condition, a, b)	Conditional result: if([value] != 0, 1, 0).
ifs(c1, v1, c2, v2, default)	Multiple ordered conditions.
Abs, Sign	Absolute value and sign.
Min, Max	Limits or selection: Min(Max([value], 0), 100).
Round, Floor, Ceiling, Truncate	Numeric rounding. Example: Round([value], 2).
Pow, Sqrt, Exp, Ln, Log, Log10	Powers and logarithms.
Sin, Cos, Tan, Asin, Acos, Atan, Atan2	Trigonometry.

Function	Purpose / example
Bit(value, mask)	Returns 1 when any masked bit is set; otherwise 0.
Bcd(value)	Converts the absolute integer digits to packed BCD. Bcd(123) produces hexadecimal 0x123.
Mod(left, right)	Remainder; returns 0 when the divisor is 0.
in(value, ...)	Returns whether value matches one of the following arguments.

Formula patterns

Goal	Expression
Pass through	[value]
Scale and offset	[value] * 100 + 5
Boolean normalization	if([value] != 0, 1, 0)
Invert boolean	if([value] == 0, 1, 0)
Clamp to 0–100	Min(Max([value], 0), 100)
Two decimals	Round([value], 2)
Test bit 3	Bit([value], 8)
Last two digits	Mod([value], 100)
Combine item values	[rpm] / Max([maxRpm], 1)

Case: Use the standard NCalc function casing shown in this manual. The custom Bit, Bcd and Mod function names are accepted case-insensitively.

Division: Protect divisors that may be zero, for example [rpm] / Max([maxRpm], 1). Test boundary values, negative values and missing dependencies before live use.

Install the FFairLink Arduino library

The FFair_Link distribution contains the source library and an installation ZIP. The current library manual version is 0.9.0 even though the desktop application is v0.8.1.

1. Open Arduino IDE.
2. Choose Sketch > Include Library > Add .ZIP Library.
3. Select FFairLink-0.9.0.zip from the distribution.
4. If replacing an older manually installed copy, remove or rename the old library outside Arduino IDE, then restart the IDE.
5. Open one of the included examples and compile it for the intended board before changing the sketch.

Library responsibilities

- Non-blocking Serial and UDP message processing.
- Device identity and handshake responses.
- Optional/required CRC8 receive validation and outgoing CRC8 generation.
- Linked integer, float, text and two-value inputs and outputs.
- Change thresholds, minimum transmission intervals and heartbeat.
- Connection state and diagnostic counters.

Sketch/library match: Installing the library does not automatically change an existing sketch. Confirm its device ID, keys, transport and API usage before uploading it to a controller assigned to a profile.

Program an Arduino-compatible controller

Core model

```
#include <FFairLink.h>

FFairLinkSerial ffair("PANEL01", "1.0");
FFairLinkInt bus(ffair, "BUS");
FFairLinkOutputInt panelSwitch(ffair, "SW", 1, 20, 1000);

void setup() {
  pinMode(2, INPUT_PULLUP);
  pinMode(LED_BUILTIN, OUTPUT);
  ffair.begin(Serial, 115200);
}

void loop() {
  ffair.update();
  digitalWrite(LED_BUILTIN, bus != 0 ? HIGH : LOW);
  panelSwitch = digitalRead(2) == LOW ? 1 : 0;
}
```

Create the transport and linked variables globally. Call ffair.update() near the start of every loop. The function processes all currently available input, handshake requests and pending linked outputs without deliberately waiting.

Input classes

Class	Stored data	Wire example
FFairLinkInt	Signed 32-bit integer	RPM=2450
FFairLinkFloat	32-bit float	TEMP=23.75
FFairLinkText<N>	Text up to sketch capacity N	MODE=AUTO
FFairLinkIntPair	Two signed integers	C=2=1
FFairLinkFloatPair	Two floats	POS=40.4=-3.7

Reading input state

```
float current = rpm.value();

if (rpm.available()) {
  // At least one valid value has been received.
}

if (rpm.changed()) {
  // The flag is returned and cleared.
}
```

Output classes and rate control

```
FFairLinkOutputFloat throttle(
  ffair, "THR",
  0.01f,  // minimum change
  50,     // minimum time between sends, ms
  1000    // heartbeat, ms; zero disables
);
```

Linked outputs retain the latest assigned value. The first value is transmitted after connection, real changes are sent when rate limits allow, and assigning the same value produces no traffic unless heartbeat or forceNext() requires a refresh. A new handshake also forces retained outputs to be published.

Callbacks and direct sends

onChange callbacks execute inside update(); keep them short and never delay or block. Direct ffair.send() and sendPair() calls bypass linked-output minChange, minTimeMs and heartbeat, so use them for deliberate events rather than continuously assigned state.

Serial COM workflow

Protocol and configuration

- FFair_Link uses 115200 baud for Serial devices and discovery.
- Use newline-terminated ASCII messages. Outgoing CRC8 is enabled by default in the library and recommended in the application.
- Keep ffair.update() running frequently. Long delay(), blocking loops or slow callbacks can make the link appear intermittent.
- Only one process should own a COM port. Close Arduino Serial Monitor before FFair_Link connects.

Message examples

```
BUS=1*XX
TEMP=23.75*XX
POS=40.4168=-3.7038*XX
```

XX represents the two hexadecimal CRC8 characters. The application can also accept incoming messages without a CRC, but invalid CRC-bearing messages are rejected. Configure the library receive mode as disabled, optional or required according to the endpoint contract.

COM-port procedure

1. Upload the sketch and close Serial Monitor.
2. Use Windows Device Manager or FFair_Link Find to identify the controller.
3. Confirm the profile Device ID matches the sketch and select the COM port.
4. Connect the device and check the handshake/version in the log.
5. Use the device transmission log to verify RX, TX and CRC before diagnosing simulator mappings.

Board reset: Opening a port can reset some Arduino boards. FFair_Link normally keeps DTR off and discovery first tries to identify a running controller without reset; allow time for a board that does reset to start its sketch.

UDP workflow

Automatic discovery

The controller configures its own Ethernet or Wi-Fi interface. FFairLink manages the FFair_Link peer. With a blank IP field, FFair_Link broadcasts a targeted handshake on the configured UDP port; the board learns the sender IP/port and uses it for subsequent messages.

```
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <FFairLink.h>
#include <FFairLinkUDP.h>

EthernetUDP udp;
FFairLinkUDP ffair("PANEL01", "1.0");

void setup() {
  Ethernet.begin(mac);
  ffair.begin(udp, 49000);
  ffair.setConnectionTimeout(3000);
}

void loop() {
  Ethernet.maintain();
  ffair.update();
}
```

Network rules

- Use a unique MAC address for every Ethernet controller.
- Confirm the board obtained a valid address; 0.0.0.0 is not usable.
- The board and PC should normally be on the same LAN. Discovery broadcasts usually do not cross routers or VLANs.
- Allow FFair_Link through Windows Firewall on the active network profile.
- Use the same UDP port in the sketch and device JSON; the bundled examples use 49000.
- Use a fixed endpoint only when routing or network policy requires it and both IP/port values are controlled.

Connection state

setConnectionTimeout(3000) makes isConnected() false when no valid handshake or message is received for three seconds. A timeout is useful for UDP and bidirectional devices; it may be unnecessary for a deliberately one-way Serial endpoint.

Reliability, performance and commissioning

Library workload

Calling ffair.update() hundreds or thousands of times per second is normal. Work per call depends mainly on bytes received and registered linked outputs. The library stores the latest value rather than a queue of every key; the shared receive buffer is 128 bytes. On an Arduino Uno, approximate object costs are 15 bytes for FFairLinkInt, 34 bytes for FFairLinkOutputInt and 28 bytes for FFairLinkText<16>, excluding transport libraries.

Traffic control

- Use minChange to suppress noise and tiny analog movements.
- Use minIntervalMs to prevent a fast source from monopolizing Serial or UDP traffic.
- Use heartbeat only where state recovery is valuable; it is not required for every signal.
- Keep keys concise and avoid duplicating the same simulator source at unnecessarily high poll rates.
- Keep Arduino callbacks and loop code non-blocking. Replace long delay() calls with elapsed-time logic.

Commissioning checklist

1. Verify power rails, common ground, input protection and actuator supplies before USB/network connection.
2. Compile the exact sketch and record its device ID, version, transport and keys.
3. Validate handshake and CRC with no simulator actions enabled.
4. Validate incoming simulator values using Test and the transmission log.
5. Validate physical inputs using RX messages before enabling actions.
6. Enable one rule at a time and verify safe simulator behavior.
7. Exercise reconnects: controller first, application first and simulator first.
8. Run a realistic-duration test and check for CRC, overflow, syntax, unknown-key and disconnect errors.
9. Back up the proven profile folder and source sketch.

Troubleshooting

Symptom	Checks

Symptom	Checks
Serial device not found	Close Serial Monitor; confirm 115200 baud; check Device ID; call update() continuously; use the correct COM port; inspect handshake response and DTR/reset timing.
Connected to wrong board	Enable handshake validation and give every controller a unique Device ID.
UDP device not found	Check link LEDs and board IP; confirm port 49000 (or configured value), same LAN, unique MAC, blank host for discovery and Windows Firewall permission.
Input never available	Check exact key, 1–8 character key rule, value count, CRC mode and diagnostics: crcErrors, valueErrors and unknownKeys.
Output never sent	Assign a value; confirm isConnected; inspect minChange/minTime; remember identical assignments do not send without heartbeat; try forceNext().
MSFS offset wrong	Verify address and signed/unsigned width; check units/scaling; test raw source before the formula.
LVar missing	Load the aircraft fully; verify FSUIPC WASM/WAPI and FSUIPC_WAPID.dll; browse the live LVar list.
X-Plane action fails	Verify Web API connection; target spelling; writable dataref; command hold/repetition settings; aircraft-specific availability.
Formula error	Check brackets, decimal dot, function casing, missing item dependencies, divide-by-zero and whether the result is numeric.
Repeated toggle / oscillation	Use explicit states or edge/pulse inputs; review firstMatch/allMatches and continuously true conditions.

Diagnostic counters in the library

```
ffair.messagesReceived();
ffair.crcErrors();
ffair.overflowErrors();
ffair.parseErrors();
ffair.keyErrors();
ffair.valueErrors();
ffair.unknownKeys();
ffair.droppedMessages();
ffair.lastRxMs();
```

Clear statistics only after recording the evidence. A rising counter is more useful than a single final number because it identifies which fault class is active during the test.

Reference

Wire protocol limits

Item	Limit / behavior
Key	1–8 ASCII letters, digits or underscore.
Values	One or two values separated by =.
Numeric token	Up to 12 characters; invalid, overflow, NaN and infinity are rejected.
Receive line buffer	128 bytes.
Text	Sketch-defined capacity up to 115 characters; printable ASCII without * or =.
CRC8	Optional on receive by default; enabled on transmit by default.
Float precision	Arduino float provides about seven significant decimal digits.

Arduino library API summary

Category	API
Transports	FFairLinkSerial, FFairLinkUDP
Inputs	FFairLinkInt, FFairLinkFloat, FFairLinkText<N>, FFairLinkIntPair, FFairLinkFloatPair
Outputs	FFairLinkOutputInt, FFairLinkOutputFloat, FFairLinkOutputText<N>, FFairLinkOutputIntPair, FFairLinkOutputFloatPair
Main loop	ffair.update()
Direct transmit	send(), sendPair(), sendPayload()
Connection	setConnectionTimeout(), isConnected()
CRC	setCrcMode(), setSendCrc()

Minimal profile agreement

Layer	Must agree
Profile device	deviceId, transport, COM/UDP endpoint, handshake setting, CRC preference.
Profile variables	Payload keys for Sim → Device and input keys for Device → Sim.

Layer	Must agree
Sketch	Compiled device ID/version, linked variable keys, transport and port/baud.
Simulator	Actual aircraft offset/LVar/control or dataref/command and writable/readable behavior.

Sample minimum contracts

Example	Device ID	Keys	Transport
MSFS minimum	MINIMUM01	BUS (to board), PB (to simulator)	Serial 115200
X-Plane minimum Serial	MINIMUM01	BUS (to board), PB (to simulator)	Serial 115200
X-Plane minimum UDP	MINIMUM01_UDP	BUS (to board), PB (to simulator)	UDP 49000

Glossary

Term	Definition
CRC8	Eight-bit cyclic redundancy check appended as *XX to detect message corruption.
DTR	Serial control line that may reset or wake some boards when a port opens.
FSUIPC offset	Typed memory-like interface exposed by FSUIPC for simulator data and control.
LVar	Aircraft Local Variable accessed through the FSUIPC WASM/WAPI path.
Dataref	X-Plane data reference used to read or, when permitted, write simulator state.
Command	X-Plane action that can be activated, held and released.
Heartbeat	Periodic retransmission of unchanged state for recovery/confirmation.
Handshake	Identity exchange used to associate a physical controller with a profile device.

Version note

This edition was prepared from the FFair_Link v0.8.1 application sources, the bundled example profiles/sketches and the FFairLink Arduino library v0.9.0 guide. Recheck the changelog and included examples after upgrading either component.