Table of Contents

Table of Contents

1 Altitude (PX4)

Download this subcategory as PDF file

PX4 reports two altitude types - AMSL (Above Mean Sea Level) and AHL (Above Home Location). Altitude AHL is displayed as Raw altitude in client.

Vehicle altitude AMSL is calculated from Raw altitude as:
Vehicle altitude AMSL = Takeoff altitude + Reported altitude AHL

If Takeoff altitude is not specified in UgCS then:
Vehicle altitude AMSL = Reported altitude AMSL

Current altitude AGL (Above Ground Level) is calculated as:
Vehicle altitude AGL = Vehicle altitude AMSL - Terrain elevation AMSL at vehicle location

2 Autopilot parameters (PX4)

Download this subcategory as PDF file

There is a number of MAVlink parameters which are changed on the autopilot during operation. Parameters are set during route upload and command execution.

Parameter	Description
COM_LOW_BAT_ACT	Modified if route parameters sets failsafe action on low battery.
MIS_YAWMODE	Set to 0 on route upload if "autoheading" is set to yes. See also Force heading to next WP
MPC_XY_CRUISE	Modified when Click&Go command issued on the vehicle.
MPC_XY_VEL_MAX	Modified when Click&Go command issued and specified speed exceeds current value of MPC_XY_VEL_MAX.
NAV_RCL_ACT	Modified if route parameters sets failsafe action on RC signal loss.
RTL_RETURN_ALT	Modified if route parameters sets Emergency return altitude.

3 Command execution specifics (Microdrones)

Download this subcategory as PDF file

Command	Support	Notes
ARM	Partial	The command issuing does not turn on motors but allows doing it from RC without battery re-inserting. It also activates telemetry sending.
DISARM	Partial	Does not stop motors but activates uplink channel and deactivates telemetry sending.
AUTOMODE	No	
MANUALMODE	No	
CLICK & GO	No	
JOYSTICK	No	
HOLD	No	
CONTINUE	No	
RETURN HOME	No	
TAKEOFF	No	
LAND	No	
EMERGENCYLAND	No	
CAMERA_TRIGGER	No	

4 Command execution specifics (Mikrokopter)

Download this subcategory as PDF file

Command	Support	Notes
ARM	No	
DISARM	No	
AUTOMODE	No	
MANUALMODE	No	
CLICK & GO	No	
JOYSTICK	No	
HOLD	No	
CONTINUE	No	
RETURN HOME	Yes	Current mission is erased in the device. It should be uploaded again in order to run it. Works only when the drone is in AUTO mode.
TAKEOFF	No	
LAND	No	
EMERGENCYLAND	No	
CAMERA_TRIGGER	No	

Use the RC to execute commands and switch between flight modes according to the manufacturer instructions.

5 Command execution specifics (PX4)

Download this subcategory as PDF file

Command	Support	Notes
ARM	Yes	Arms vehicle.
DISARM	Yes	Disarms vehicle.
AUTOMODE	Yes	Start mission from first waypoint. Sets vehicle into Mission flight mode.
MANUALMODE	Yes	Sets Manual mode.
CLICK & GO	Yes	Sets Click & Go (single waypoint) mode.
JOYSTICK	No	Vehicle control via joystick.
HOLD	Yes	Pause mission execution. The drone will loiter at its current position.
CONTINUE	Yes	Continue with mission execution from next waypoint. Works from Manual and Click&Go modes.
RETURN HOME	Yes	Vehicle will return to home location. See also Home Location (HL) support.
TAKEOFF	Yes	
LAND	Yes	
EMERGENCYLAND	Yes	
CAMERA_TRIGGER	No	
DRIECT_PAYLOAD_CONTROL	Yes	Payload control via keyboard/joystick.

6 Command shading (PX4)

Download this subcategory as PDF file

UGCS Client can show command buttons in different shades. It is always possible to press all buttons disregarding of shade. Highlighted buttons suggest recommended commands, depending on vehicle current status.

7 Common configuration file parameters (Microdrones)

Download this subcategory as PDF file

VSM configuration file is a text file specified via command line argument - -config of the VSM application.
? **Example:**`--config /etc/opt/ugcs/vsm-ardupilot.conf`

Each configuration parameter is defined as a line in the configuration file with the following structure: `name1.name2....nameX = value`
where name1, name2 ... nameX are arbitrary names separated by dots to divide a variable into logical blocks and a value which can be a number value or a text string depending on the context. See below the description about common VSM configuration parameters.

7.1 UgCS server configuration

VSM can connect to UgCS in two different ways:
? Listen for connection form the UgCS server. See Listening address and Listening port. When VSM is configured in listening mode automatic VSM discovery can be set up, too. See Automatic service discovery
? Initiate connection to UgCS server. See UgCS server address and UgCS server port.
At least one of the above must be configured for VSM to work.

Listening address
Optional.
? **Name:** ucs.local_listening_address = [IP address]
? **Description:** Local address to listen for incoming connections from UgCS server.
? **Default:** 0.0.0.0 (listen on all local addresses)
? **Example:** `ucs.local_listening_address = 10.0.0.2`

Listening port
Optional.
? **Name:** ucs.local_listening_port = [port number]
? **Description:** Local TCP port to listen for incoming connections from UgCS server.
? **Example:** `ucs.local_listening_port = 5556`

UgCS server address.
Optional.
? **Name:** ucs.address = [IP address]
? **Description:** UgCS server address to connect to.
? **Example:** `ucs.address = 1.2.3.4`

UgCS server port.
Optional.
? **Name:** ucs.port = [port number]
? **Description:** UgCS server port.
? **Example:** `ucs.port = 3335`

Retry timeout.
Optional.
? **Name:** ucs.retry_timeout = [seconds]
? **Description:** Retry timeout for outgoing server connections in seconds.
? **Default:** 10
? **Example:** `retry_timeout = 11`

7.2 Automatic service discovery

VSM can respond to automatic service discovery requests form UgCS server. When this parameter is not configured, service discovery is disabled.
Optional.
? **Name:** service_discovery.vsm_name = [Service name]
? **Description:** Human readable service name.
? **Example:** `service_discovery.vsm_name = Ardupilot VSM`

7.3 Logging configuration

Level.
Optional.
? **Name:** log.level = [error|warning|info|debug]
? **Description:** Logging level.
? **Default:** info
? **Example:** `log.level = debug`

File path.
Optional.
? **Name:** log.file_path = [path to a file]
? **Description:** Absolute or relative (to the current directory) path to a logging file. Logging is disabled if logging file is not defined. File should be writable. Backslash should be escaped with a backslash.
? **Example:** `log.file = /var/opt/ugcs/log/vsm-ardupilot/vsm-ardupilot.log`
? **Example:** `log.file = C:\\Users\\John\\AppData\\Local\\UGCS\\logs\\vsm-ardupilot\\vsm-ardupilot.log`

Maximum single file size.
Optional.
? **Name:** log.single_max_size = [size]
? **Description:** Maximum size of a single log file. When maximum size is exceeded, existing file is renamed by adding a time stamp and logging is continued into the empty file. [size] should be defined as a number postfixed by a case insensitive multiplier:

? Gb, G, Gbyte, Gbytes: for Giga-bytes
? Mb, M, Mbyte, Mbytes: for Mega-bytes
? Kb, K, Kbyte, Kbytes: for Kilo-bytes
? no postfix: for bytes
? **Default:** 100 Mb
? **Example:** `log.single_max_size = 500 Mb`

Maximum number of old log files.
Optional.
? **Name:** log.max_file_count = [number]
? **Description:** Log rotation feature. Maximum number of old log files to keep. After reaching single_max_size of current log file, VSM will rename it with current time in extension and start new one. VSM will delete older logs so the number of old logs does not exceed the max_file_count.
? **Default:** 1
? **Example:** `log.max_file_count = 5`

7.4 Mission dump path

Optional.
? **Name:** [prefix].mission_dump_path = [path to a file]
? **Description:** File to dump all generated missions to. Timestamp is appended to the name. Delete the entry to disable mission dumping. All directories in the path to a file should be already created.
`vehicle.ardupilot.mission_dump_path = C:\\tmp\\ardupilot_dump`

7.5 Command execution control

When vehicle is connected via unreliable link VSM will retry each command several times before failing. This section describes the parameters which control the command execution.

Command try count.
? **Name:** vechicle.command_try_count = <number of="" times>="">
? **Description:** Number of times the command will be issued before declaring it as failed. Must be greater than zero.
? **Default:** 3
? **Example:** `vechicle.command_try_count = 5`

Command timeout.
? **Name:** vechicle.command_timeout = <timeout in="" seconds>="">
? **Description:** Time to wait for response on issued command before retrying.
? **Unit:** second
? **Default:** 1

Vehicle serial prefix.
Optional.
? **Name:** vehicle.serial_prefix = string
? **Description:** String value used as prefix for all vehicle serial numbers conected to this VSM. Can be used to connect vehicles with equal serial numbers to the same server via different VSMs
? **Default:** not defined
? **Example:** `vehicle.serial_prefix = group1:`

8 Common configuration file parameters (Mikrokopter)

Download this subcategory as PDF file

VSM configuration file is a text file specified via command line argument - -config of the VSM application.
Example: `--config /etc/opt/ugcs/vsm-ardupilot.conf`

Each configuration parameter is defined as a line in the configuration file with the following structure: `name1.name2....nameX = value`
where name1, name2 ... nameX are arbitrary names separated by dots to divide a variable into logical blocks and a value which can be a number value or a text string depending on the context. See below the description about common VSM configuration parameters.

8.1 UgCS server configuration

VSM can connect to UgCS in two different ways:
? Listen for connection form the UgCS server. See Listening address and Listening port. When VSM is configured in listening mode automatic VSM discovery can be set up, too. See Automatic service discovery
? Initiate connection to UgCS server. See UgCS server address and UgCS server port.
At least one of the above must be configured for VSM to work.

Listening address
Optional.
? **Name:** ucs.local_listening_address = [IP address]
? **Description:** Local address to listen for incoming connections from UgCS server.
? **Default:** 0.0.0.0 (listen on all local addresses)
? **Example:** `ucs.local_listening_address = 10.0.0.2`

Listening port
Optional.
? **Name:** ucs.local_listening_port = [port number]
? **Description:** Local TCP port to listen for incoming connections from UgCS server.
? **Example:** `ucs.local_listening_port = 5556`

UgCS server address.
Optional.
? **Name:** ucs.address = [IP address]
? **Description:** UgCS server address to connect to.
? **Example:** `ucs.address = 1.2.3.4`

UgCS server port.
Optional.
? **Name:** ucs.port = [port number]
? **Description:** UgCS server port.
? **Example:** `ucs.port = 3335`

Retry timeout.
Optional.
? **Name:** ucs.retry_timeout = [seconds]
? **Description:** Retry timeout for outgoing server connections in seconds.
? **Default:** 10
? **Example:** `retry_timeout = 11`

8.2 Automatic service discovery

VSM can respond to automatic service discovery requests form UgCS server. When this parameter is not configured, service discovery is disabled.
Optional.
? **Name:** service_discovery.vsm_name = [Service name]
? **Description:** Human readable service name.
? **Example:** service_discovery.vsm_name = Ardupilot VSM

8.3 Logging configuration

Level.
Optional.
? **Name:** log.level = [error|warning|info|debug]
? **Description:** Logging level.
? **Default:** info
? **Example:** `log.level = debug`

File path.
Optional.
? **Name:** log.file_path = [path to a file]
? **Description:** Absolute or relative (to the current directory) path to a logging file. Logging is disabled if logging file is not defined. File should be writable. Backslash should be escaped with a backslash.
? **Example:** `log.file = /var/opt/ugcs/log/vsm-ardupilot/vsm-ardupilot.log`
? **Example:** `log.file = C:\\Users\\John\\AppData\\Local\\UGCS\\logs\\vsm-ardupilot\\vsm-ardupilot.log`

Maximum single file size.
Optional.
? **Name:** log.single_max_size = [size]
? **Description:** Maximum size of a single log file. When maximum size is exceeded, existing file is renamed by adding a time stamp and logging is

continued into the empty file. [size] should be defined as a number postfixed by a case insensitive multiplier:
? Gb, G, Gbyte, Gbytes: for Giga-bytes
? Mb, M, Mbyte, Mbytes: for Mega-bytes
? Kb, K, Kbyte, Kbytes: for Kilo-bytes
? no postfix: for bytes
? **Default:** 100 Mb
? **Example:** `log.single_max_size = 500 Mb`

Maximum number of old log files.
Optional.
? **Name:** log.max_file_count = [number]
? **Description:** Log rotation feature. Maximum number of old log files to keep. After reaching single_max_size of current log file, VSM will rename it with current time in extension and start new one. VSM will delete older logs so the number of old logs does not exceed the max_file_count.
? **Default:** 1
? **Example:** `log.max_file_count = 5`

8.4 Mission dump path

Optional.
? **Name:** [prefix].mission_dump_path = [path to a file]
? **Description:** File to dump all generated missions to. Timestamp is appended to the name. Delete the entry to disable mission dumping. All directories in the path to a file should be already created.
? **Example:** `vehicle.ardupilot.mission_dump_path = C:\\tmp\\ardupilot_dump`

8.5 Command execution control

When vehicle is connected via unreliable link VSM will retry each command several times before failing. This section describes the parameters which control the command execution.

Command try count
? **Name:** vechicle.command_try_count = <number of="" times>="">
? **Description:** Number of times the command will be issued before declaring it as failed. Must be greater than zero.
? **Default:** 3
? **Example:** `vechicle.command_try_count = 5`

Command timeout
? **Name:** vechicle.command_timeout = <timeout in="" seconds>="">
? **Description:** Time to wait for response on issued command before retrying.
? **Unit:** second
? **Default:** 1

Vehicle serial prefix.
Optional.
? **Name:** vehicle.serial_prefix = string
? **Description:** String value used as prefix for all vehicle serial numbers conected to this VSM. Can be used to connect vehicles with equal serial numbers to the same server via different VSMs
? **Default:** not defined
? **Example:** `vehicle.serial_prefix = group1:`

9 Common configuration file parameters (PX4)

Download this subcategory as PDF file

VSM configuration file is a text file specified via command line argument - -config of the VSM application.
? **Example:** `--config /etc/opt/ugcs/vsm-ardupilot.conf`

Each configuration parameter is defined as a line in the configuration file with the following structure: `name1.name2....nameX = value`
where name1, name2 ... nameX are arbitrary names separated by dots to divide a variable into logical blocks and a value which can be a number value or a text string depending on the context. See below the description about common VSM configuration parameters.

9.1 UgCS server configuration

VSM can connect to UgCS in two different ways:
? Listen for connection form the UgCS server. See Listening address and Listening port. When VSM is configured in listening mode automatic VSM discovery can be set up, too. See Automatic service discovery
? Initiate connection to UgCS server. See UgCS server address and UgCS server port.
At least one of the above must be configured for VSM to work.

Listening address
Optional.
? **Name:** ucs.local_listening_address = [IP address]
? **Description:** Local address to listen for incoming connections from UgCS server.
? **Default:** 0.0.0.0 (listen on all local addresses)
? **Example:** `ucs.local_listening_address = 10.0.0.2`

Listening port
Optional.
? **Name:** ucs.local_listening_port = [port number]
? **Description:** Local TCP port to listen for incoming connections from UgCS server.
? **Example:** `ucs.local_listening_port = 5556`

UgCS server address.
Optional.
? **Name:** ucs.address = [IP address]
? **Description:** UgCS server address to connect to.
? **Example:** `ucs.address = 1.2.3.4`

UgCS server port.
Optional.
? **Name:** ucs.port = [port number]
? **Description:** UgCS server port.
? **Example:** `ucs.port = 3335`

Retry timeout.
Optional.
? **Name:** ucs.retry_timeout = [seconds]
? **Description:** Retry timeout for outgoing server connections in seconds.
? **Default:** 10
? **Example:** `retry_timeout = 11`

9.2 Automatic service discovery

VSM can respond to automatic service discovery requests form UgCS server. When this parameter is not configured, service discovery is disabled.
Optional.
? **Name:** service_discovery.vsm_name = [Service name]
? **Description:** Human readable service name.
? **Example:** `service_discovery.vsm_name = Ardupilot VSM`

9.3 Logging configuration

Level.
Optional.
? **Name:** log.level = [error|warning|info|debug]
? **Description:** Logging level.
? **Default:** info
? **Example:** `log.level = debug`

File path.
Optional.
? **Name:** log.file_path = [path to a file]
? **Description:** Absolute or relative (to the current directory) path to a logging file. Logging is disabled if logging file is not defined. File should be writable. Backslash should be escaped with a backslash.
? **Example:** `log.file = /var/opt/ugcs/log/vsm-ardupilot/vsm-ardupilot.log`
? **Example:** `log.file = C:\\Users\\John\\AppData\\Local\\UGCS\\logs\\vsm-ardupilot\\vsm-ardupilot.log`

Maximum single file size.
Optional.
? **Name:** log.single_max_size = [size]
? **Description:** Maximum size of a single log file. When maximum size is exceeded, existing file is renamed by adding a time stamp and logging is continued into the empty file. [size] should be defined as a number postfixed by a case insensitive multiplier:

? Gb, G, Gbyte, Gbytes: for Giga-bytes
? Mb, M, Mbyte, Mbytes: for Mega-bytes
? Kb, K, Kbyte, Kbytes: for Kilo-bytes
? no postfix: for bytes
? **Default:** 100 Mb
? **Example:** `log.single_max_size = 500 Mb`

Maximum number of old log files.
Optional.
? **Name:** log.max_file_count = [number]
? **Description:** Log rotation feature. Maximum number of old log files to keep. After reaching single_max_size of current log file, VSM will rename it with current time in extension and start new one. VSM will delete older logs so the number of old logs does not exceed the max_file_count.
? **Default:** 1
? **Example:** `log.max_file_count = 5`

9.4 Mission dump path

Optional.
? **Name:** [prefix].mission_dump_path = [path to a file]
? **Description:** File to dump all generated missions to. Timestamp is appended to the name. Delete the entry to disable mission dumping. All directories in the path to a file should be already created.
? **Example:** `vehicle.ardupilot.mission_dump_path = C:\\tmp\\ardupilot_dump`

9.5 Command execution control

When vehicle is connected via unreliable link VSM will retry each command several times before failing. This section describes the parameters which control the command execution.

Command try count
? **Name:** vechicle.command_try_count = <number of="" times>="">
? **Description:** Number of times the command will be issued before declaring it as failed. Must be greater than zero.
? **Default:** 3
? **Example:** `vechicle.command_try_count = 5`

Command timeout
? **Name:** vechicle.command_timeout = <timeout in="" seconds>="">
? **Description:** Time to wait for response on issued command before retrying.
? **Unit:** second
? **Default:** 1

Vehicle serial prefix.
Optional.
? **Name:** vehicle.serial_prefix = string
? **Description:** String value used as prefix for all vehicle serial numbers conected to this VSM. Can be used to connect vehicles with equal serial numbers to the same server via different VSMs
? **Default:** not defined
? **Example:** `vehicle.serial_prefix = group1:`

10 Communication with devices (Microdrones)

Download this subcategory as PDF file

VSM can communicate with Vehicle over different communication channels Currently supported channels are below:

10.1 Serial port configuration

VSM which communicates with vehicles via serial ports should define at least one serial port, otherwise VSM will not try to connect to the vehicles. Port name and baud rate should be both defined.

Port name.
Required.
? **Name:** connection.serial.[index].name = [regular expression]
? **Description:** Ports which should be used to connect to the vehicles by given VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Expression is case insensitive on Windows. [index] is a arbitrary port indexing name.
? **Example:** `connection.serial.1.name = /dev/ttyUSB[0-9]+|com[0-9]+`
? **Example:** `connection.serial.2.name = com42`

Port baud rate.
Required.
? **Name:** connection.serial.[index].baud.[baud index] = [baud]
? **Description:** Baud rate for port opening. [baud index] is an optional arbitrary name used when it is necessary to open the same serial port using multiple baud rates. [index] is an arbitrary port indexing name.
? **Example:** `connection.serial.1.baud.1 = 9600`
? **Example:** `connection.serial.1.baud.2 = 57600`
? **Example:** `connection.serial.2.baud = 38400`

Excluded port name.
Optional.
? **Name:** connection.serial.exclude.[exclude index] = [regular expression]
? **Description:** Ports which should not be used for vehicle access by this VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Filter is case insensitive on Windows. [exclude index] is a arbitrary indexing name used when more than one exclude names are defined.
? **Example:** `connection.serial.exclude.1 = /dev/ttyS.?`
? **Example:** `connection.serial.exclude = com1`

Serial port arbiter.
Optional.
? **Name:** connection.serial.use_arbiter = [yes|no]
? **Description:** Enable (yes) or disable (no) serial port access arbitration between VSMs running on the same machine. It is recommended to have it enabled to avoid situation when multiple VSMs try to open the same port simultaneously.
? **Default:** yes
? **Example:** `connection.serial.use_arbiter = no`

10.2 Outgoing TCP connection configuration

VSM can be configured to connect to the vehicle via TCP. VSM will try to establish connection to the specified address:port.
Used to connect to vehicle simulator or when vehicle is equipped with WiFi adapter.

Remote TCP port.
Required.
? **Name:** connection.tcp_out.[index].port = [port number]
? **Description:** Remote port to connect to.
? **Example:** `connection.tcp_out.1.port = 5762`

IP-address for outgoing TCP connection.
Required.
? **Name:** connection.tcp_out.[index].address = [IP-address]
? **Description:** IP-address of vehicle to connect to.
? **Example:** `connection.tcp_out.1.address = 10.0.0.111`

10.3 Incoming TCP connection configuration

VSM can be configured to listen for incoming TCP connections from the vehicle. Multiple vehicles are supported on the same port.
Used to connect to vehicle equipped with WiFi adapter.

Local listening TCP port.
Required.
? **Name:** connection.tcp_in.[index].local_port = [port number]
? **Description:** Remote port to connect to.
? **Example:** `connection.tcp_in.1.local_port = 5762`

Local IP address.
Optional.
? **Name:** connection.tcp_in.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (all interfaces)
? **Example:** `connection.tcp_in.1.local_address = 127.0.0.1`

10.4 Outgoing UDP connection configuration

VSM can be configured to connect to the vehicle via UDP. VSM will try to establish UDP connection to the specified address:port.

Remote IP-address for UDP.
Required.
? **Name:** connection.udp_out.[index].address = [IP-address]
? **Description:** Remote IP-address to send outgoing UDP packets to.
? **Example:** `connection.udp_out.1.address = 192.168.1.1`

Remote UDP port.
Required.
? **Name:** connection.udp_out.[index].port = [port number]
? **Description:** Remote UDP port to send outgoing packets to.
? **Example:** `connection.udp_out.1.port = 14551`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_out.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_out.1.local_address = 0.0.0.0`

Local UDP port.
Optional.
? **Name:** connection.udp_out.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Default:** 0 (bind to random port)
? **Example:** `connection.udp_out.1.local_port = 14550`

10.5 Incoming UDP connection configuration

VSM can be configured to listen for UDP conncections from the vehicle. Vehicle must be actively sending heartbeat/telemetry on specified UDP port before it can be detected by VSM. VSM will automatically detect multiple vehicles on the same port. This is very useful for "drone swarm" setups as there is no need to specify connector for each vehicle and no need to know the IP address of each vehicle in advance.

Local UDP port.
Required.
? **Name:** connection.udp_in.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Example:** `connection.udp_in.1.local_port = 14550`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_in.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_in.1.local_address = 0.0.0.0`

10.6 Incoming UDP connection configuration (any peer)

This connection type is similar to "udp_in" with the exception that all incoming traffic will be recevied as one stream. It is used for special purpose connections and cannot be used to connect vehicles.

Local UDP port.
Required.
? **Name:** connection.udp_any.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Example:** `connection.udp_any.1.local_port = 14550`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_any.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_any.1.local_address = 0.0.0.0`

10.7 Named pipes

VSM is able to communicate with vehicle over named pipe. The pipe must already exist.

? **Name:** connection.pipe.[index].port = pipe_name
? **Description:** Pipe name
? **Example:** `connection.pipe.1.name = \\pipe\my_named_pipe`

10.8 Proxy configuration

VSM is able to communicate with vehicle via proxy service which redirects dataflow received from vehicle through TCP connection to VSM and vice versa using specific protocol. In other words, proxy service appears as a router between vehicle and VSM. At the moment there is one implementation of proxy in UgCS called XBee Connector, which retranslates data from ZigBee network to respective VSM.

IP-address for proxy.
Required.
? **Name:** connection.proxy.[index].address = [IP-address]
? **Description:** IP-address to connect proxy to. Specify local or remote address.
? **Example:** `connection.proxy.1.address = 127.0.0.1`

TCP port for proxy.
Required.
? **Name:** connection.proxy.[index].port = [port number]
? **Description:** TCP port to be connected with proxy through. Should be the same as in congiguration on proxy side.
? **Example:** `connection.proxy.1.port = 5566`

11 Communication with devices (Mikrokopter)

Download this subcategory as PDF file

VSM can communicate with Vehicle over different communication channels Currently supported channels are below:

11.1 Serial port configuration

VSM which communicates with vehicles via serial ports should define at least one serial port, otherwise VSM will not try to connect to the vehicles. Port name and baud rate should be both defined.

Port name.
Required.
? **Name:** connection.serial.[index].name = [regular expression]
? **Description:** Ports which should be used to connect to the vehicles by given VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Expression is case insensitive on Windows. [index] is a arbitrary port indexing name.
? **Example:** `connection.serial.1.name = /dev/ttyUSB[0-9]+|com[0-9]+`
? **Example:** `connection.serial.2.name = com42`

Port baud rate.
Required.
? **Name:** connection.serial.[index].baud.[baud index] = [baud]
? **Description:** Baud rate for port opening. [baud index] is an optional arbitrary name used when it is necessary to open the same serial port using multiple baud rates. [index] is an arbitrary port indexing name.
? **Example:** `connection.serial.1.baud.1 = 9600`
? **Example:** `connection.serial.1.baud.2 = 57600`
? **Example:** `connection.serial.2.baud = 38400`

Excluded port name.
Optional.
? **Name:** connection.serial.exclude.[exclude index] = [regular expression]
? **Description:** Ports which should not be used for vehicle access by this VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Filter is case insensitive on Windows. [exclude index] is a arbitrary indexing name used when more than one exclude names are defined.
? **Example:** `connection.serial.exclude.1 = /dev/ttyS.?`
? **Example:** `connection.serial.exclude = com1`

Serial port arbiter.
Optional.
? **Name:** connection.serial.use_arbiter = [yes|no]
? **Description:** Enable (yes) or disable (no) serial port access arbitration between VSMs running on the same machine. It is recommended to have it enabled to avoid situation when multiple VSMs try to open the same port simultaneously.
? **Default:** yes
? **Example:**`connection.serial.use_arbiter = no`

11.2 Outgoing TCP connection configuration

VSM can be configured to connect to the vehicle via TCP. VSM will try to establish connection to the specified address:port.
Used to connect to vehicle simulator or when vehicle is equipped with WiFi adapter.

Remote TCP port.
Required.
? **Name:** connection.tcp_out.[index].port = [port number]
? **Description:** Remote port to connect to.
? **Example:** `connection.tcp_out.1.port = 5762`

IP-address for outgoing TCP connection.
Required.
? **Name:** connection.tcp_out.[index].address = [IP-address]
? **Description:** IP-address of vehicle to connect to.
? **Example:** `connection.tcp_out.1.address = 10.0.0.111`

11.3 Incoming TCP connection configuration

VSM can be configured to listen for incoming TCP connections from the vehicle. Multiple vehicles are supported on the same port.
Used to connect to vehicle equipped with WiFi adapter.

Local listening TCP port.
Required.
? **Name:** connection.tcp_in.[index].local_port = [port number]
? **Description:** Remote port to connect to.
? **Example:** connection.tcp_in.1.local_port = 5762

Local IP address.
Optional.
? **Name:** connection.tcp_in.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (all interfaces)
? **Example:** `connection.tcp_in.1.local_address = 127.0.0.1`

11.4 Outgoing UDP connection configuration

VSM can be configured to connect to the vehicle via UDP. VSM will try to establish UDP connection to the specified address:port.

Remote IP-address for UDP.
Required.
? **Name:** connection.udp_out.[index].address = [IP-address]
? **Description:** Remote IP-address to send outgoing UDP packets to.
? **Example:** `connection.udp_out.1.address = 192.168.1.1`

Remote UDP port.
Required.
? **Name:** connection.udp_out.[index].port = [port number]
? **Description:** Remote UDP port to send outgoing packets to.
? **Example:** `connection.udp_out.1.port = 14551`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_out.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_out.1.local_address = 0.0.0.0`

Local UDP port.
Optional.
? **Name:** connection.udp_out.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Default:** 0 (bind to random port)
? **Example:** `connection.udp_out.1.local_port = 14550`

11.5 Incoming UDP connection configuration

VSM can be configured to listen for UDP conncections from the vehicle. Vehicle must be actively sending heartbeat/telemetry on specified UDP port before it can be detected by VSM. VSM will automatically detect multiple vehicles on the same port. This is very useful for "drone swarm" setups as there is no need to specify connector for each vehicle and no need to know the IP address of each vehicle in advance.

Local UDP port.
Required.
? **Name:** connection.udp_in.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Example:** `connection.udp_in.1.local_port = 14550`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_in.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_in.1.local_address = 0.0.0.0`

11.6 Incoming UDP connection configuration (any peer)

This connection type is similar to "udp_in" with the exception that all incoming traffic will be recevied as one stream. It is used for special purpose connections and cannot be used to connect vehicles.

Local UDP port.
Required.
? **Name:** connection.udp_any.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Example:** `connection.udp_any.1.local_port = 14550`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_any.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_any.1.local_address = 0.0.0.0`

11.7 Named pipes

VSM is able to communicate with vehicle over named pipe. The pipe must already exist.

? **Name:** connection.pipe.[index].port = pipe_name
? **Description:** Pipe name
? **Example:** `connection.pipe.1.name = \\pipe\my_named_pipe`

11.8 Proxy configuration

VSM is able to communicate with vehicle via proxy service which redirects dataflow received from vehicle through TCP connection to VSM and vice versa using specific protocol. In other words, proxy service appears as a router between vehicle and VSM. At the moment there is one implementation of proxy in UgCS called XBee Connector, which retranslates data from ZigBee network to respective VSM.

IP-address for proxy.
Required.
? **Name:** connection.proxy.[index].address = [IP-address]
? **Description:** IP-address to connect proxy to. Specify local or remote address.
? **Example:** `connection.proxy.1.address = 127.0.0.1`

TCP port for proxy.
Required.
? **Name:** connection.proxy.[index].port = [port number]
? **Description:** TCP port to be connected with proxy through. Should be the same as in congiguration on proxy side.
? **Example:** `connection.proxy.1.port = 5566`

12 Communication with devices (PX4)

Download this subcategory as PDF file

VSM can communicate with Vehicle over different communication channels Currently supported channels are below:

12.1 Serial port configuration

VSM which communicates with vehicles via serial ports should define at least one serial port, otherwise VSM will not try to connect to the vehicles. Port name and baud rate should be both defined.

Port name.
Required.
? **Name:** connection.serial.[index].name = [regular expression]
? **Description:** Ports which should be used to connect to the vehicles by given VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Expression is case insensitive on Windows. [index] is a arbitrary port indexing name.
? **Example:** `connection.serial.1.name = /dev/ttyUSB[0-9]+|com[0-9]+`
? **Example:** `connection.serial.2.name = com42`

Port baud rate.
Required.
? **Name:** connection.serial.[index].baud.[baud index] = [baud]
? **Description:** Baud rate for port opening. [baud index] is an optional arbitrary name used when it is necessary to open the same serial port using multiple baud rates. [index] is an arbitrary port indexing name.
? **Example:** `connection.serial.1.baud.1 = 9600`
? **Example:** `connection.serial.1.baud.2 = 57600`
? **Example:** `connection.serial.2.baud = 38400`

Excluded port name.
Optional.
? **Name:** connection.serial.exclude.[exclude index] = [regular expression]
? **Description:** Ports which should not be used for vehicle access by this VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Filter is case insensitive on Windows. [exclude index] is a arbitrary indexing name used when more than one exclude names are defined.
? **Example:** `connection.serial.exclude.1 = /dev/ttyS.?`
? **Example:** `connection.serial.exclude = com1`

Serial port arbiter.
Optional.
? **Name:** connection.serial.use_arbiter = [yes|no]
? **Description:** Enable (yes) or disable (no) serial port access arbitration between VSMs running on the same machine. It is recommended to have it enabled to avoid situation when multiple VSMs try to open the same port simultaneously.
? **Default:** yes
`connection.serial.use_arbiter = no`

12.2 Outgoing TCP connection configuration

VSM can be configured to connect to the vehicle via TCP. VSM will try to establish connection to the specified address:port.
Used to connect to vehicle simulator or when vehicle is equipped with WiFi adapter.

Remote TCP port.
Required.
? **Name:** connection.tcp_out.[index].port = [port number]
? **Description:** Remote port to connect to.
? **Example:** connection.tcp_out.1.port = 5762

IP-address for outgoing TCP connection.
Required.
? **Name:** connection.tcp_out.[index].address = [IP-address]
? **Description:** IP-address of vehicle to connect to.
? **Example:** connection.tcp_out.1.address = 10.0.0.111

12.3 Incoming TCP connection configuration

VSM can be configured to listen for incoming TCP connections from the vehicle. Multiple vehicles are supported on the same port.
Used to connect to vehicle equipped with WiFi adapter.

Local listening TCP port.
Required.
? **Name:** connection.tcp_in.[index].local_port = [port number]
? **Description:** Remote port to connect to.
? **Example:** connection.tcp_in.1.local_port = 5762

Local IP address.
Optional.
? **Name:** connection.tcp_in.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (all interfaces)
? **Example:** connection.tcp_in.1.local_address = 127.0.0.1

12.4 Outgoing UDP connection configuration

VSM can be configured to connect to the vehicle via UDP. VSM will try to establish UDP connection to the specified address:port.

Remote IP-address for UDP.
Required.
? **Name:** connection.udp_out.[index].address = [IP-address]
? **Description:** Remote IP-address to send outgoing UDP packets to.
? **Example:** `connection.udp_out.1.address = 192.168.1.1`

Remote UDP port.
Required.
? **Name:** connection.udp_out.[index].port = [port number]
? **Description:** Remote UDP port to send outgoing packets to.
? **Example:** `connection.udp_out.1.port = 14551`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_out.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_out.1.local_address = 0.0.0.0`

Local UDP port.
Optional.
? **Name:** connection.udp_out.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Default:** 0 (bind to random port)
? **Example:** `connection.udp_out.1.local_port = 14550`

12.5 Incoming UDP connection configuration

VSM can be configured to listen for UDP conncections from the vehicle. Vehicle must be actively sending heartbeat/telemetry on specified UDP port before it can be detected by VSM. VSM will automatically detect multiple vehicles on the same port. This is very useful for "drone swarm" setups as there is no need to specify connector for each vehicle and no need to know the IP address of each vehicle in advance.

Local UDP port.
Required.
? **Name:** connection.udp_in.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Example:**`connection.udp_in.1.local_port = 14550`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_in.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_in.1.local_address = 0.0.0.0`

12.6 Incoming UDP connection configuration (any peer)

This connection type is similar to "udp_in" with the exception that all incoming traffic will be recevied as one stream. It is used for special purpose connections and cannot be used to connect vehicles.

Local UDP port.
Required.
? **Name:** connection.udp_any.[index].local_port = [port number]
? **Description:** Local UDP port to listen for incoming packets on.
? **Example:** `connection.udp_any.1.local_port = 14550`

Local IP-address for UDP.
Optional.
? **Name:** connection.udp_any.[index].local_address = [IP-address]
? **Description:** Local ip address to bind to.
? **Default:** 0.0.0.0 (bind to all interfaces)
? **Example:** `connection.udp_any.1.local_address = 0.0.0.0`

12.7 Named pipes

VSM is able to communicate with vehicle over named pipe. The pipe must already exist.

? **Name:** connection.pipe.[index].port = pipe_name
? **Description:** Pipe name
? **Example:**`connection.pipe.1.name = \\pipe\my_named_pipe`

12.8 Proxy configuration

VSM is able to communicate with vehicle via proxy service which redirects dataflow received from vehicle through TCP connection to VSM and vice versa using specific protocol. In other words, proxy service appears as a router between vehicle and VSM. At the moment there is one implementation of proxy in UgCS called XBee Connector, which retranslates data from ZigBee network to respective VSM.

IP-address for proxy.
Required.
? **Name:** connection.proxy.[index].address = [IP-address]
? **Description:** IP-address to connect proxy to. Specify local or remote address.
? **Example:**`connection.proxy.1.address = 127.0.0.1`

TCP port for proxy.
Required.
? **Name:** connection.proxy.[index].port = [port number]
? **Description:** TCP port to be connected with proxy through. Should be the same as in congiguration on proxy side.
? **Example:** `connection.proxy.1.port = 5566`

13 Configuration file (Microdrones)

Download this subcategory as PDF file

Default configuration file of the PX4 VSM suits most needs and it is generally not necessary to modify it.

Configuration file location:

? **Microsoft Windows:**
C:\Program Files (x86)\UgCS\bin\vsm-microdrones.conf

? **GNU/Linux:**
/etc/opt/ugcs/vsm-microdrones.conf

? **Apple OS X:**
/Users/[user name]/Library/Application Support/UGCS/configuration/vsm-microdrones.conf

13.1 Serial port configuration

Typically vehicle is connected to UgCS via radio datalink which appears as serial port when USB cable is plugged in. See Serial port configuration for details.

? Example: `connection.serial.1.name = COM21 connection.serial.1.baud = 38400`

13.2 Common parameters

All VSMs share a common set of configuration file parameters described in Common configuration file parameters. MikroKopter VSM configuration file prefix is: *vehicle.microdrones*

13.3 Model name override

By default the VSM sets ?MD? model name for the Microdrones vehicles. It can be overridden to more specific name in the VSM configuration.

? Example:
```
vehicle.microdrones.custom.my_drone.serial_number = 1102
vehicle.microdrones.custom.my_drone.model_name = MD4-200
```

In this example the model name for the drone with serial number ?1102? is overridden to value ?MD4-200?.

14 Configuration file (Mikrokopter)

Download this subcategory as PDF file

Default configuration file of the PX4 VSM suits most needs and it is generally not necessary to modify it.

Configuration file location:

? **Microsoft Windows:**
C:\Program Files (x86)\UgCS\bin\vsm-mikrokopter.conf

? **GNU/Linux:**
/etc/opt/ugcs/vsm-mikrokopter.conf

? **Apple OS X:**
/Users/[user name]/Library/Application Support/UGCS/configuration/vsm-mikrokopter.conf

14.1 WP-event value

This parameter defines which value will be sent in wp-event-value field of the waypoint with "take photo" action. See Camera trigger action section.

? Example: `vehicle.mikrokopter.wp_event_value = 50`

14.2 Serial port configuration

Typically vehicle is connected to UgCS via radio datalink which appears as serial port when USB cable is plugged in. See Serial port configuration for details.

? Example: `connection.serial.1.name = COM21 connection.serial.1.baud = 57600`

14.3 Common parameters

All VSMs share a common set of configuration file parameters described in Common configuration file parameters. MikroKopter VSM configuration file prefix is: *vehicle.mikrokopter*

15 Configuration file (PX4)

Download this subcategory as PDF file

Default configuration file of the PX4 VSM suits most needs and it is generally not necessary to modify it.

Configuration file location:

? **Microsoft Windows:**
C:\Program Files (x86)\UgCS\bin\vsm-px4.conf

? **GNU/Linux:**
/etc/opt/ugcs/vsm-px4.conf

? **Apple OS X:**
/Users/[user name]/Library/Application Support/UGCS/configuration/vsm-px4.conf

15.1 Common parameters

All VSMs share a common set of configuration file parameters described in Common configuration file parameters. PX4 VSM configuration file prefix is:

vehicle.px4

15.2 Communication channel configuration

There must be at least one communication channel defined, otherwise VSM will not try to connect to the vehicle. See Communication with devices for details.

Default installation is configured to detect autopilot automatically on any available serial port at 57600 or 115Kbps.

15.3 Model name and serial number override

Optional.

? Name: vehicle.px4.custom.[name].system_id = [system id]

? Name: vehicle.px4.custom.[name].model_name = [model name]

15.4 Configuration file

? **Name:** vehicle.px4.custom.[name].serial_number = [serial number]

? **Description:** In UgCS each vehicle is identified by a unique combination of model name and serial number represented as text strings. By default, PX4 vehicles are identified with a model name PX4 and serial number equal with the Mavlink system id read from the vehicle. It can be overridden by these parameters, where [name] is an arbitrary vehicle name, [system id] is the original Mavlink system id which should be overridden, [model name] is a new model name to be visible to the UgCS, [serial number] is a new serial number to be visible to the UgCS.

? **Example:**

```
vehicle.px4.custom.my_drone.system_id = 2
vehicle.px4.custom.my_drone.model_name = PX4Quad
vehicle.px4.custom.my_drone.serial_number = 123456
```

15.5 Camera trigger type

There are different ways to control camera payload in px4 family. First one is "common" one and utilizes SET_SERVO or REPEAT_SERVO commands. Camera trigger control is linked with autopilot servo output and triggering take place when we send proper amount of PWM signal in that output.

Second one is when we use high level commands MAV_CMD_IMAGE_START_CAPTURE and MAV_CMD_IMAGE_STOP_CAPTURE to control triggering.
So we introduce parameter

```
vehicle.px4.camera_trigger_type
```

When it set to 0, VSM will use high level commands MAV_CMD_IMAGE_START_CAPTURE and MAV_CMD_IMAGE_STOP_CAPTURE And when it set to 1 - VSM will use MAV_CMD_DO_REPEAT_SERVO command along with camera_servo_idx, camera_servo_pwm and camera_servo_time parameter values.

```
Note: Yuneec 520 with payload always override this parameter to 0 value.
```

15.6 Telemetry rate configuration

PX4 VSM supports setting custom telemetry rates based on mavlink message type. There are 8 message types which are used by VSM to get the essential state info from vehicle: SYS_STATUS, GLOBAL_POSITION_INT, ATTITUDE, VFR_HUD, GPS_RAW_INT, ALTITUDE, HEARTBEAT, HOME_POSITION.

Other messages which are sent by autopilot are disabled by VSM to save datalink channel bandwidth.

It is possible to configure rate for each message type separately.

? Required: No.

? Supported values: 0.1 - 50.0

? Default: 2

? Description: Message count per second.

? Example:

```
vehicle.px4.telemetry_rate.ALTITUDE = 0.5
vehicle.px4.telemetry_rate.ATTITUDE = 0.5
vehicle.px4.telemetry_rate.GLOBAL_POSITION_INT = 0.5
vehicle.px4.telemetry_rate.GPS_RAW_INT = 0.5
vehicle.px4.telemetry_rate.HEARTBEAT = 0.5
vehicle.px4.telemetry_rate.HOME_POSITION = 0.5
vehicle.px4.telemetry_rate.SYS_STATUS = 0.5
vehicle.px4.telemetry_rate.VFR_HUD = 0.5
```

15.7 Mavlink version

PX4 VSM supports MAVLink versions 1 and 2 and detects the correct version automatically. In setups when there is some intermediate device between VSM and autopilot this detection can fail. For those scenarios it is possible to force mavlink version used by VSM.

? Required: No.

? Supported values: 1 2 auto

? Default: auto

? Description: 1: always use mavlink version 1, 2: always use mavlink version 2, auto: Autodetect mavlink version.

? Example:

```
vehicle.px4.mavlink_protocol_version = 1
```

15.8 Force heading to next WP

By default VSM will automatically generate commands for vehicle to set heading towards next waypoint. This behavior can be disabled by setting parameter "autoheading" to no. See also Heading behavior

? Required: No.

? Supported values: yes, no

? Default: yes

? Description:

no - do not change heading between waypoints. This disables override of parameter MIS_YAWMODE on mission upload. Vehicle heading will depend on MIS_YAWMODE. If MIS_YAWMODE is set to 1 (next WP) then all heading actions in mission will be ignored.

yes - change heading towards next waypoint. When set, each mission upload sets MIS_YAWMODE to zero (yaw controlled by mission)

? Example:

```
vehicle.px4.autoheading = no
```

15.9 Mavlink message injection

Ardupilot VSM can receive mavlink packets and forward them to the vehicle if vehicle with specified target_id is connected. It can be used to send GPS RTK corrections to vehicles. If message has no target_id or traget_id is 0 then it is sent to all connected vehicles. Supported messages are: COMMAND_LONG, COMMAND_INT, GPS_INJECT_DATA and GPS_RTCM_DATA. The prefix mavlink_injection supports all the same syntax as "connection" prefix.

? Required: No.

? Supported values: Same as those for connection prefix.

? Default: Not set.

? Example: `mavlink_injection.udp_any.1.local_port = 44444`

15.10 Mavlink System ID

MAVlink System ID used for outgoing MAVlink messages.

? Required: No.

? Supported values: 1 - 254

? Default: 1.

? Example:

```
mavlink.vsm_system_id = 100
```

16 Connecting UgCS and PX4

Download PX4 VSM User Guide as a PDF book

16.1 Connecting UgCS and PX4

17 Connection to PX4 simulator (PX4)

Download this subcategory as PDF file

PX4 VSM can be configured to connect to PX4 simulator via UDP connection on port 14550. Add this line to *vsm-px4.conf* file:

```
connection.udp_in.local_port = 14550
```

Please refer to PX4 documentation on how to install and configure SITL (Software In The Loop) simulation.

Warning

PX4 simulator must be launched before PX4 VSM is launched. Otherwise, if VSM is already running it will terminate the VSM when launched. This applies only to the case when VSM is running on the same host as simulation.

18 Connection using ZigBee interface (PX4)

Download this subcategory as PDF file

There is a possibility to connect UgCS to PX4 vehicle using ZigBee interface. Connection is performed with two or more Digi XBee ZigBee modules (one on ground side, others on vehicles side) and dedicated UgCS software component called XBee Connector. Please refer to XBee Connector user guide for details.

In order to use this kind of connection, disable Serial port configuration and enable Proxy configuration.

19 Fail-safe actions (Microdrones)

Download this subcategory as PDF file

GPS Lost:

Action	Result
Wait	Aircraft tries to maintain altitude
Land	Aircraft slowly descends

RC Lost:

Action	Result
Wait	Aircraft returns home and lands
Land	Aircraft returns home and lands
Return Home	Aircraft changes altitude to failsafe ALT(50m) and returns home
Continue	Aircraft continues mission

Battery Low:

Action	Result
Wait	If possible aircraft returns home and lands, if not possible slowly descends
Land	If possible aircraft returns home and lands, if not possible slowly descends
Return Home	If possible aircraft returns home and lands, if not possible slowly descends
Continue	Aircraft continues mission

20 Fail-safe actions (Mikrokopter)

Download this subcategory as PDF file

Fail-safe actions can be set only in KopterTool.

21 Fail-safe actions (PX4)

Download this subcategory as PDF file

GPS Lost:
This parameter is not supported. Vehicle will behave as specified. By default it will wait for 30s. If GPS signal does not restore vehicle will land.

RC Lost:

Action	Result
Wait	Aircraft changes altitude to failsafe altitude and returns home
Land	Aircraft lands even if in loiter mode
Return Home	Aircraft changes altitude to failsafe altitude and returns home
Continue	Aircaft continues mission

Battery Low:

Action	Result
Land	Aircraft changes altitude to failsafe altitude and returns home
Return Home	Aircraft changes altitude to failsafe altitude and returns home
Continue	Aircaft continues mission

22 First time vehicle connection (Microdrones)

Download this subcategory as PDF file

See Disclaimer.

Please follow these steps to connect an Microdrones vehicle to the UgCS:

1. Microdrones vehicle must be properly configured, calibrated and tested using tools and instruction from the official Microdrones web site prior to using it with UgCS. UgCS does not support initial configuration, setup and calibration of Microdrones vehicles.

2. Turn on the vehicle. There are two communication channels in a drone - uplink and downlink. Uplink channel is connected via USB-serial cable, plug it in the drone connector (connect to FC, not NC - connector flat side to the battery, see manufacturer manual for detailed explanation) and to the computer where VSM is running. The uplink channel can be used for mission uploading and commands execution. For telemetry reception downlink channel is needed which is connected to the downlink wire on the drone bottom. Use manufacturer provided downlink equipment which uses video transmitter sound channel for telemetry data transferring and downlink decoder ground unit. Alternatively, can connect own USB-serial cable or radio-modem to the downlink wire. Before that, switch the telemetry output to digital mode using a jumper on the drone board, see manufacturer manual for the details.

For initial vehicle set up in UgCS, can connect either uplink or downlink whichever is more convenient.

For all connection options there will be USB-serial device on the PC side so proper OS driver for virtual serial port should be installed. Please refer to the communication equipment manufacturer documentation about driver installation instructions.

3. Open Vehicles window in UgCS Client and wait until new vehicle appears there automatically. Either Uplink or Downlink connection should be available. Select the necessary vehicle and click Edit to start editing the corresponding vehicle profile. Now change the default vehicle name to the convenient one:

4. Repeat steps above for each Microdrones vehicle.

Note that if both uplink and downlink are connected only uplink will be active because the Microdrones stops telemetry sending after receptio

The downlink can be activated either by issuing "ARM" command or by disconnecting uplink and re-inserting a battery.

23 First time vehicle connection (Mikrokopter)

Download this subcategory as PDF file

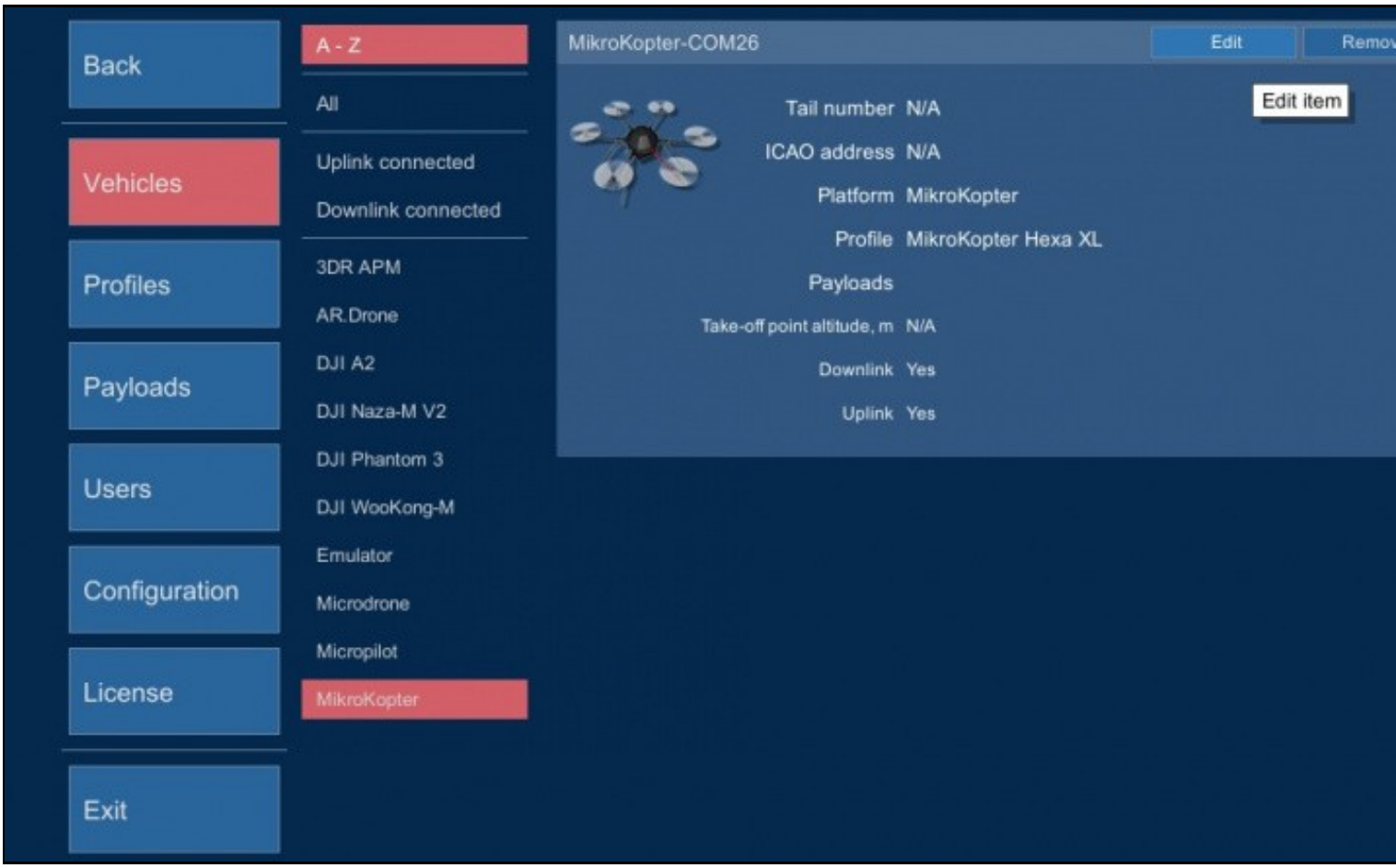
See Disclaimer.

Please follow these steps to connect an MikroKopter vehicle to the UgCS:

1. MikroKopter vehicle must be properly configured, calibrated and tested using tools and instruction from the official MikroKopter web site prior to using it with UgCS. UgCS does not support initial configuration, setup and calibration of MikroKopter vehicles.

2. Turn on the vehicle and plug in the radio modem paired with the vehicle or direct USB cable from the MikroKopter board to the computer where VSM is running. UgCS uses serial ports for communication with MikroKopter vehicles. Serial interface based communication devices like WI232 radio modems (and their analogs) and direct USB-serial connections are supported, as long as OS driver for virtual serial port is installed and serial port is successfully created. Please refer to the communication equipment manufacturer documentation about driver installation instructions.

3. Open Vehicles window in UgCS Client and wait until new vehicle appears there automatically. Both Uplink and Downlink connections should be available. Choose the corresponding vehicle for editing by clicking on the menu item and selecting Edit button. Now select the vehicle profile and change the default vehicle name to the convenient one:



Vehicle profile needs to be assigned to allow mission planning with this vehicle. Vehicle avatar should be assigned in vehicle profile to properly see the vehicle location on map.

4. Repeat steps above for each MikroKopter vehicle.

Please note that there is no technical possibility to distinguish between different MK drones if they are connected via the same port. The only way to identify drones is looking at their serial port instances. To ensure permanent drones mapping, please make sure, each drone is connected via dedicated serial port.

This firstly means that it is not possible to share the same radio-modem or USB-serial cable between different drones. If using Windows OS it usually reserves unique serial port number for each USB-serial device. Linux OS requires additional measures - manually provide udev rules for permanent device mapping based on its serial number (if such exists. Some USB devices do not have serial number specified, and it is not possible to use them to connect multiple MikroKopters). Udev rules creation example
If using own serial device naming on Linux, do not forget to add the corresponding name to the VSM configuration:

```
connection.serial.2.name = /dev/mikrokopter_1 connection.serial.2.baud.1 = 57600
```

Supported MikroKopter firmware versions:

? 0.x

? 2.00 may be supported but not tested.

24 First time vehicle connection(PX4)

Download this subcategory as PDF file

See Disclaimer.

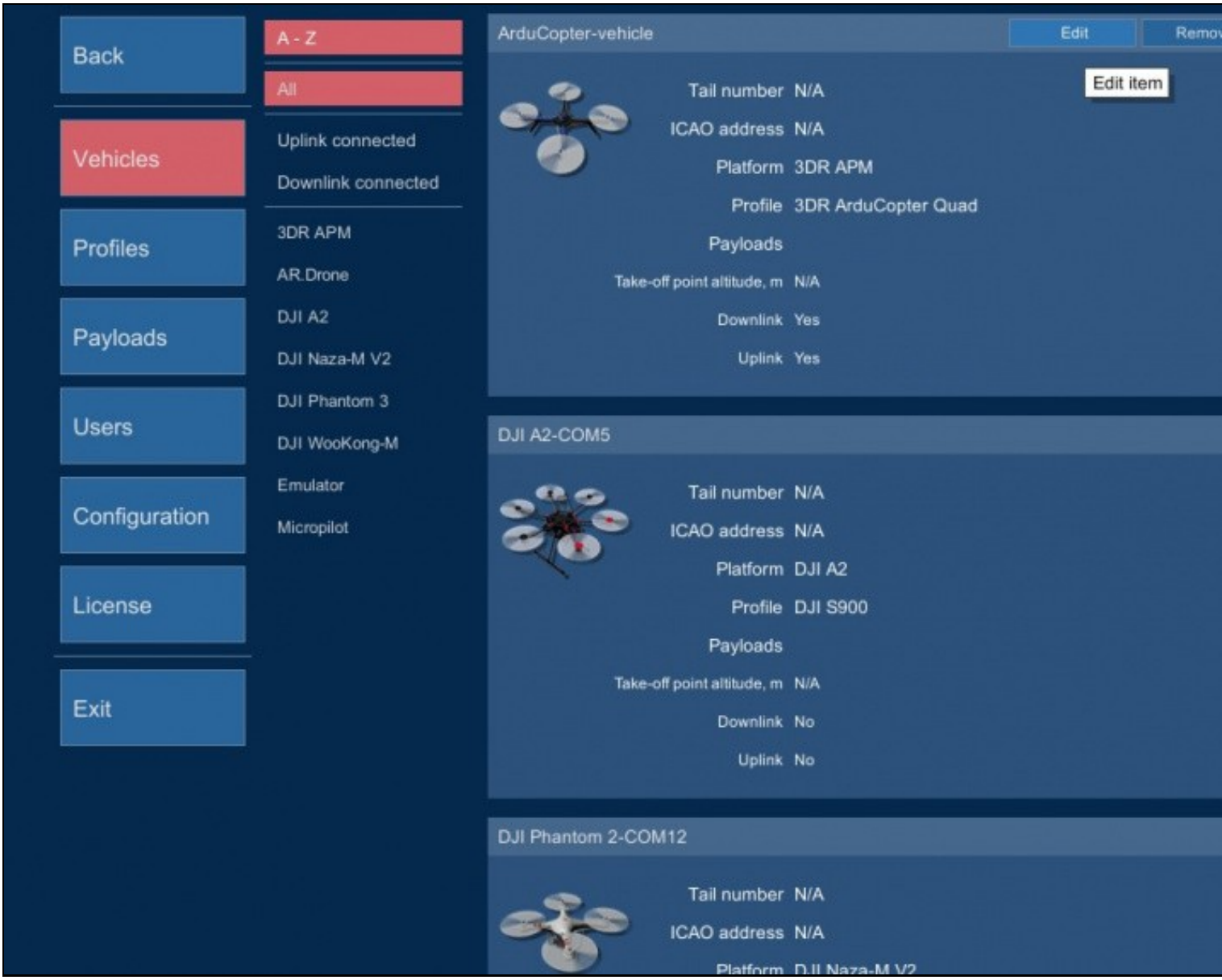
Please follow these steps to connect an PX4 vehicle to the UgCS:

1. PX4 vehicle must be properly configured, calibrated and tested using tools and instruction from the official PX4 web site prior to using it with UgCS. UgCS does not support initial configuration, setup and calibration of PX4 driven vehicles.

2. If more than one PX4 vehicle is planned to be used with UgCS, it must be ensured that each vehicle has a unique system id as defined by the parameter SYSID_THISMAV, otherwise UgCS will not be able to distinguish between different vehicles and it will not be possible to operate vehicles normally. To change the parameter, please use the official PX4 configuration software like QGroundControl.

3. Turn on the vehicle and plug in the radio modem paired with the vehicle or direct USB cable from the PX4 board to the computer where VSM is running. UgCS uses serial ports for communication with PX4 vehicles. Standard communication devices like 3DR radio modems (and their analogs) and direct USB connections are supported, as long as OS driver for virtual serial port is installed and serial port is successfully created. Please refer to the communication equipment manufacturer documentation about driver installation instructions.

4. As soon as uplink and downlink connection is established, the vehicle should appear in the active vehicles list in main window. Open Vehicles window from main menu and choose the corresponding vehicle for editing by clicking on the menu item and selecting Edit button. Select the vehicle profile and change the default vehicle name to the convenient one:



Vehicle profile needs to be assigned to allow mission planning with this vehicle. Vehicle avatar should be assigned in vehicle profile to properly see the vehicle location on map.

5. Repeat steps above for each PX4 vehicle.

Supported vehicle types:

? Copters

? VTOL vehicles

Supported PX4 firmware versions:

? 1.7+

25 Home Location (HL) support (PX4)

Download this subcategory as PDF file

PX4 does not support setting HL from UgCS. HL is always automatically set to the current location when vehicle is armed. UgCS is able to read the current HL form the vehicle and is display it on the map.

Warning

Route waypoint altitudes are calculated based on HL altitude. Thus it is crucially important to launch the vehicle from the planned HL. When operating in area with variable terrain altitude make sure the home location is specified correctly in the mission before upload.

Note

Safest way to set correct HL is to set it explicitly at the current vehicle position.

25.1 Landing at Home Location

Vehicle behavior (land or do not land) after returning at Home Location depends on the autopilot configuration.

26 Mission execution specifics (Microdrones)

Download this subcategory as PDF file

Home location cannot be modified by UgCS. It is controlled by the autopilot and is permanently bound to a take-off position.

27 Mission execution specifics (Mikrokopter)

Download this subcategory as PDF file

Before executing a mission, Mikrokopter must be set to Altitude hold mode and throttle must be in middle position!

Flight plan element / action	Support	Notes
Camera control	Partial	Only pitch control is supported.
Camera trigger	Partial	Only photo shot is supported.
Panorama	Partial	Panorama is always counted from North or last set heading
Take-off	No	
Landing	Yes	
Set camera by time	No	
Set camera by distance	Yes	
POI	Yes	
Heading	Yes	
Wait	Yes	

Take-off is not supported in scope of automatic mission execution. Take-off the drone either manually or using auto-start feature of the MikroKopter which can be activated by switch on RC. See MikroKopter vendor documentation.

CareFree feature must be enabled in MikroKopter for all actions which automatically control copter heading (like POI, yaw set etc.) See manual

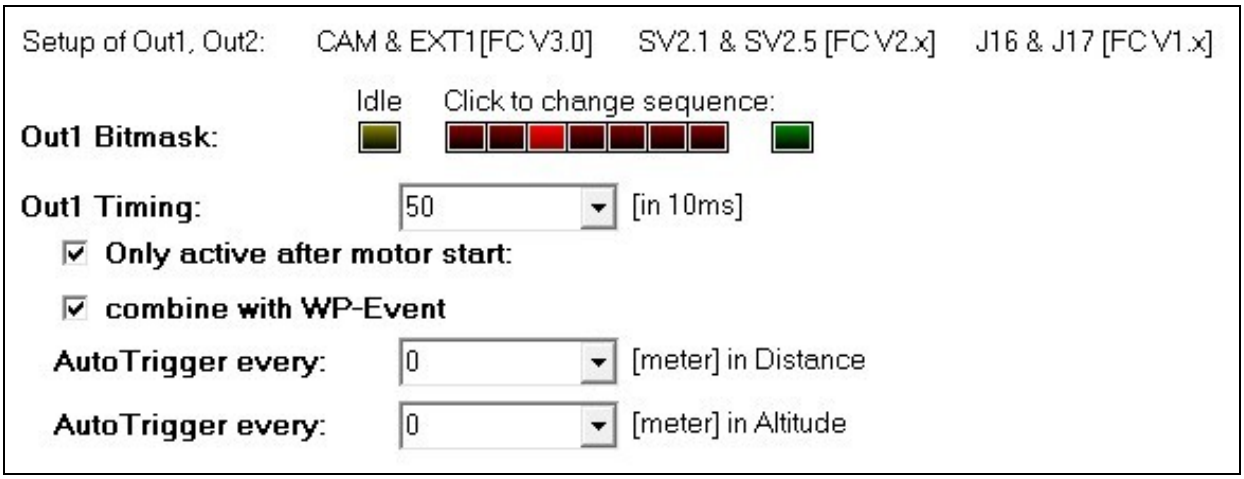
None of fail-safe mode adjustments is supported in the UgCS mission parameters (as well as emergency return altitude value). Use MikroKopter original software instead to set up fail-safe parameters.

Home position can not be changed from software. It is controlled by the autopilot hardware and always corresponds to launch position.

27.1 Camera trigger action

Can be used to trigger payload-specific action during mission execution. It triggers WP-event in the MikroKopter. See manual and manualfor more detailed description of this MK feature. As described there, before using this feature, set up pattern for the flight controller output OUT1 or SRV3 using KopterTool software. Try to use shutter cable or IR-controller provided by the drone manufacturer.

When the "take photo" action is fired the waypoint is created with wp-event-channel field set to the value specified in the VSM configuration (see WP-event value section). It is duration in deciseconds of one unit in output signal pattern. Also waypoint duration is set to time which is enough to execute at least one cycle of the pattern. Due to granularity difference in MK protocol the pattern may start the second cycle of execution before the waypoint is finished.



In KopterTool output configuration ensure to have set "Combine with WP-event" option.

28 Mission execution specifics (PX4)

Download this subcategory as PDF file

28.1 Mission action support

Flight plan element / action	Support	Notes
Change speed	Yes	See section Vehicle speed in mission below.
Wait	Yes	
Panorama	No	
Set camera mode	Yes	
Set camera by time	Yes	
Set camera by distance	Yes	
Set camera attitude	Yes	
Set POI	Yes	Starting from PX4 version 1.8
Change heading	Yes	See section Heading behavior below.

28.2 Vehicle speed in mission

Vehicle speed in mission depends not only on the speed set for each waypoint but also on maximum climb and descent rates. For best results make sure the "Max climb rate" and "Max descent rate" specified in UgCS Vehicle Profile is equal to the rates configured on the vehicle: parameters MPC_Z_VEL_MAX_UP and MPC_Z_VEL_MAX_DN respectively.

Warning

PX4 will not always fly along the straight line between waypoints if waypoints are at different altitudes.

The exact trajectory will depend on the slope angle, speed specified in mission and parameters MPC_Z_VEL_MAX_UP or MPC_Z_VEL_MAX_DN. Consider route consisting of 2 waypoints:

? WP1 at altitude 30m, speed:1m/s

? WP2 50m apart at altitude 20m.

? MPC_Z_VEL_MAX_DN is set to 5m/s

PX4 vehicle at WP1 will descend rapidly at 5m/s vertical speed while maintaining ?1m/s ground speed and then fly horizontally towards WP2.

Workaround for the problem is to use "Safe" Trajectory type. In that case UGcS will generate only vertical and horizontal segments.

28.3 Heading behavior

Vehicle heading is controlled by "Heading" waypoint actions specified in route. If Heading is not specified for the route waypoint then VSM calculates the heading automatically to point to the next WP.

The above behavior can be disabled via Force heading to next WP parameter. In that case the vehicle behavior will depend on the autopilot parameter MIS_YAWMODE.

If MIS_YAWMODE is set to zero then heading WP action will make vehicle to change heading accordingly. For waypoints which do not have explicit heading action vehicle will keep the last heading.

If MIS_YAWMODE is set to non-zero then all heading actions in uploaded route will be ignored and vehicle will always point to the next WP by default.

VTOL heading

Change yaw for VTOL vehicles works only when hovering over the waypoint. Vehicle will always fly with nose pointing to next waypoint.

For Yaw action to succeed it must be used together with "Wait" action.

28.4 Flights below Home Location

Warning

PX4 does not support flying below Home Location. Make sure all route waypoints are above HL. See also Home Location (HL) support.

29 Telemetry information specifics (Microdrones)

Download this subcategory as PDF file

Nothing specific.

30 Telemetry information specifics (Mikrokopter)

Download this subcategory as PDF file

Nothing specific.

31 Telemetry information specifics (PX4)

Download this subcategory as PDF file

31.1 Air speed

If there is no air speed sensor onboard, air speed will be shown as "Not available". If there is an air speed sensor onboard, the air speed value will be shown.

31.2 RC link quality

RC Link quality reporting is not supported.

32 Waypoint turn types (PX4)

Download this subcategory as PDF file

Turn type	Support	Notes
Straight	Yes	The vehicle will fly a straight line to the location specified as a lat, lon and altitude.
Spline	No	

33 Yuneec specific notes (PX4)

Download this subcategory as PDF file

Yuneec autopilot is based on PX4 but it lacks some features:

? Failsafe settings on RC loss and battery low are not supported.

? Speed setting during Click&Go is not supported.

? Heading change during Click&Go is not supported.