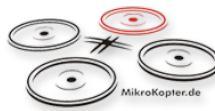


MikroKopter VSM User Guide

UgCS 3.1.871



Copyright (c) 2018, Smart Projects Holdings Ltd

Contents

1	Connecting MikroKopter autopilot to UgCS	1
1.1	First time vehicle connection	1
1.2	Mission execution specifics	2
1.2.1	Camera trigger action	2
1.3	Command execution specifics	3
1.4	Telemetry information specifics	3
1.5	Fail-safe actions	4
1.6	Configuration file	4
1.6.1	WP-event value	4
1.6.2	Serial port configuration	4
1.6.3	Common parameters	4
1.7	Common configuration file parameters	4
1.7.1	UgCS server configuration	5
1.7.2	Automatic service discovery	6
1.7.3	Logging configuration	6
1.7.4	Mission dump path	7
1.7.5	Command execution control	7
1.8	Communication with devices	7
1.8.1	Serial port configuration	8
1.8.2	Outgoing TCP connection configuration	9
1.8.3	Incoming TCP connection configuration	9
1.8.4	Outgoing UDP connection configuration	10
1.8.5	Incoming UDP connection configuration	11
1.8.6	Incoming UDP connection configuration (any peer)	11
1.8.7	Named pipes	12
1.8.8	Proxy configuration	12
2	Disclaimer	12

1 Connecting MikroKopter autopilot to UgCS

1.1 First time vehicle connection

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 your 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 you can select the vehicle profile and change the default vehicle name to be convenient for you:

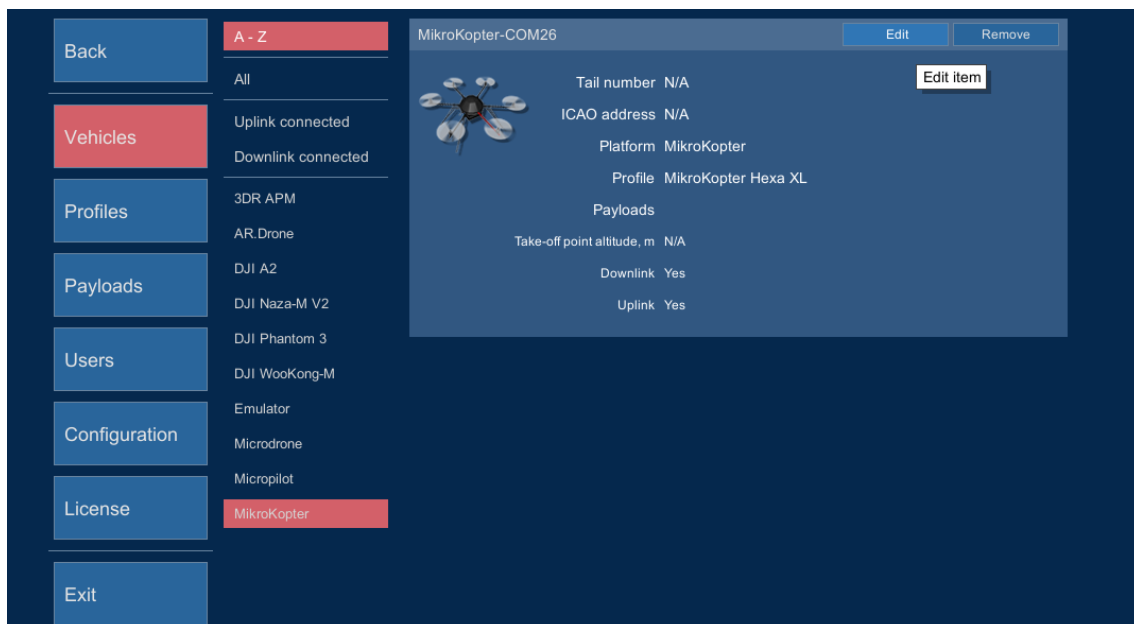


Figure 1: New MikroKopter vehicle

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 your 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 be sure that you have permanent drones mapping, please ensure that you connect each drone via a dedicated serial port.

This firstly means that you cannot share the same radio-modem or USB-serial cable between different drones. If you are using Windows OS it usually reserves unique serial port number for each USB-serial device. On Linux OS you will need to take 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, you will not be able to use them to connect multiple MikroKopters). Udev rules creation example: <http://noctis.-de/ramblings/linux/49-howto-fixed-name-for-a-udev-device.html>

If you use your own serial device naming on Linux, do not forget to add the corresponding name to the VSM configuration:

```
connection.serial.2.name = /dev/mikrokoetter_1
connection.serial.2.baud.1 = 57600
```

Supported MikroKopter firmware versions:

- 0.x
- 2.00 may be supported but not tested.

1.2 Mission execution specifics

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. You should 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 <http://www.mikrokoetter.de/ucwiki/en/CareFree>

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.

1.2.1 Camera trigger action

Can be used to trigger payload-specific action during mission execution. It triggers WP-event in the MikroKopter. See <http://www.mikrokoetter.de/ucwiki/en/WaypointEvent> and <http://www.-mikrokoetter.de/ucwiki/en/WpEvent> for more detailed description of this MK feature. As described there, before you can use this feature you should set up pattern for the flight controller output OUT1 or SRV3 using KopterTool software. You may want to use [shutter cable](#) or [IR-controller](#) provided by the drone manufacturer.

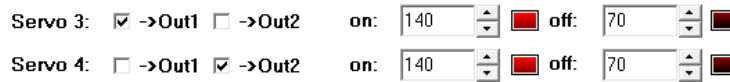


Figure 2: Servo mapping

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). Its 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.

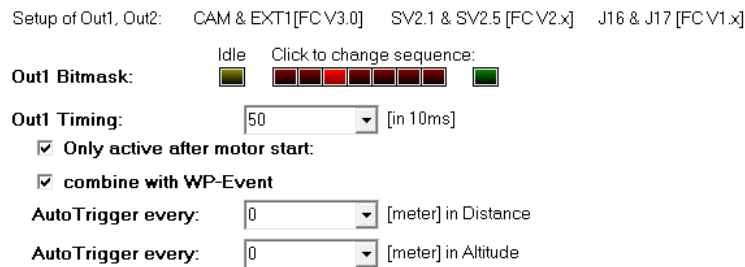


Figure 3: Example of output configuration

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

1.3 Command execution specifics

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

1.4 Telemetry information specifics

Nothing specific.

1.5 Fail-safe actions

Fail-safe actions can be set only in KopterTool.

1.6 Configuration file

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

Configuration file location:

- **On Microsoft Windows:**

```
C:\Program Files (x86)\UgCS\bin\vsm-mikrokoetter.conf
```

- **On GNU/Linux:**

```
/etc/opt/ugcs/vsm-mikrokoetter.conf
```

- **On Apple OS X:**

```
/Users/[user name]/Library/Application Support/UGCS/configuration/vsm-mikrokoetter.conf
```

1.6.1 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.

```
vehicle.mikrokoetter.wp_event_value = 50
```

1.6.2 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
```

1.6.3 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.mikrokoetter
```

1.7 Common configuration file parameters

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.

1.7.1 UgCS server configuration

VSM can connect to UgCS in two different ways:

- Listen for connection from 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.

1.7.1.1 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

1.7.1.2 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

1.7.1.3 UgCS server address

Optional.

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

1.7.1.4 UgCS server port

Optional.

- **Name:** ucs.port = [port number]
- **Description:** UgCS server port.
- **Example:** ucs.port = 3335

1.7.1.5 Retry timeout

Optional.

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

1.7.2 Automatic service discovery

VSM can respond to automatic service discovery requests from 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

1.7.3 Logging configuration

1.7.3.1 Level

Optional.

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

1.7.3.2 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`

1.7.3.3 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

1.7.3.4 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
-

1.7.4 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.
- **Example:** vehicle.ardupilot.mission_dump_path = C:\\tmp\\ardupilot_dump

1.7.5 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.

1.7.5.1 Command try count

- **Name:** vehicle.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:** vehicle.command_try_count = 5

1.7.5.2 Command timeout

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

1.8 Communication with devices

VSM can communicate with Vehicle over different communication channels

Currently supported channels:

- Serial port, see [Serial port configuration](#) for details.

- Outgoing TCP, see [Outgoing TCP connection configuration](#) for details.
- Incoming TCP, see [Incoming TCP connection configuration](#) for details.
- Outgoing UDP, see [Outgoing UDP connection configuration](#) for details.
- Incoming UDP, see [Incoming UDP connection configuration](#) for details.
- Incoming UDP (any peer), see [Incoming UDP connection configuration \(any peer\)](#) for details.
- Named pipe , see [Named pipes](#) for details.
- vsm-proxy (XBee), see [Proxy configuration](#) for details.

1.8.1 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.

1.8.1.1 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 an arbitrary port indexing name.
- **Example:** connection.serial.1.name = /dev/ttyUSB[0-9]+|com[0-9]+
- **Example:** connection.serial.2.name = com42

1.8.1.2 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

1.8.1.3 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 an arbitrary indexing name used when more than one exclude names are defined.
- **Example:** connection.serial.exclude.1 = /dev/ttyS.*
- **Example:** connection.serial.exclude = com1

1.8.1.4 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

1.8.2 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.

1.8.2.1 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

1.8.2.2 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

1.8.3 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.

1.8.3.1 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

1.8.3.2 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

1.8.4 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.

1.8.4.1 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

1.8.4.2 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

1.8.4.3 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

1.8.4.4 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

1.8.5 Incoming UDP connection configuration

VSM can be configured to listen for UDP connections from the vehicle. Vehicle must be actively sending heart-beat/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.

1.8.5.1 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

1.8.5.2 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

1.8.6 Incoming UDP connection configuration (any peer)

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

1.8.6.1 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

1.8.6.2 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

1.8.7 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

1.8.8 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.

1.8.8.1 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

1.8.8.2 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 configuration on proxy side.
- **Example:** connection.proxy.1.port = 5566

2 Disclaimer

DISCLAIMER OF WARRANTIES AND LIMITATIONS ON LIABILITY.

(a) SMART PROJECTS HOLDINGS MAKE NO REPRESENTATIONS OR WARRANTIES REGARDING THE ACCURACY OR COMPLETENESS OF ANY CONTENT OR FUNCTIONALITY OF THE PRODUCT AND ITS DOCUMENTATION.

(b) SMART PROJECTS HOLDINGS DISCLAIM ALL WARRANTIES IN CONNECTION WITH THE PRODUCT, AND WILL NOT BE LIABLE FOR ANY DAMAGE OR LOSS RESULTING FROM YOUR USE OF THE PRODUCT. INCLUDING BUT NOT LIMITED TO INJURY OR DEATH OF USER OR ANY THIRD PERSONS OR DAMAGE TO PROPERTY.

(c) THE SOFTWARE IS SUPPLIED AS IS WITH NO WARRANTIES AND CAN BE USED ONLY AT USERS OWN RISK.