SimpleBGC32 Serial API protocol specification

Applicable for all 32-bit gimbal controllers

Last updated: Jun 21, 2024

Revision history:

- rev. 0.1 - 24.03.2015: this is first revision
- rev. 0.2 – 27.03.2015: add missed data
- rev. 0.3 – 30.04.2015: add missed data in CMD_READ_PARAMS_EXT
- rev. 0.4 – 01.07.2015: extended CMD_CONTROL; add MENU_CMD_LEVEL_ROLL_PITCH; FRAME_ANGLE_XX replaced by ROTOR_ANGLE_XX in the CMD_REALTIME_DATA_4; updated CMD_AHRS_HELPER.
- rev. 0.5 – 30.07.2015: extended set PROFILE_FLAGS1, GENERAL_FLAGS1; extended set CMD_EXECUTE_MENU; deprecated FRAME_CAM_ANGLE_XX.
- rev. 0.6 – 12.08.2015: new mode for CMD_CONTROL; new commands CMD_GET_ANGLES_EXT, CMD_SET_ADJ_VARS_VAL.
- rev. 0.7 – 22.10.2015: new parameters ORDER_OF_AXES, EULER_ORDER; set of PROFILE_FLAGS1, GENERAL_FLAGS1 extended; SKIP_GYRO_CALIB options extended.
- rev. 0.8 – 09.11.2015: CMD_AHRS_HELPER is extended.
- rev. 0.9 – 22.12.2015: new command CMD_GYRO_CORRECTION; new adj. var. FRAME_HEADING_ANLGE and GYRO_HEADING_CORRECTION; extended GENERAL_FLAGS1 and PROFILE_FLAGS1.
- rev. 0.10 – 13.02.2016: updated CMD_AUTO_PID; extended range of NOTCH_GAIN.
- rev. 0.11 – 07.03.2016: new command CMD_READ_PARAMS_EXT2; new parameter MOTOR_MAG_LINK_FINE; new command CMD_CALIB_MOTOR_MAG_LINK; ACC_LIMITER split to axes; extended form of CMD_HELPER_DATA.
- rev. 0.12 – 02.04.2016: new commands CMD_DATA_STREAM_INTERVAL, CMD_REALTIME_DATA_CUSTOM.
- rev. 0.13 – 05.06.2016: new command CMD_BEEP_SOUND; new adjustment variables.
- rev. 0.14 – 21.06.2016: CMD_ADJ_VARS_STATE described.
- rev. 0.15 – 09.07.2016: extended CMD_READ_PARAMS_EXT2; extended CFG_FLAGS in CMD_AUTO_PID; new command CMD_CALIB_INFO; corrected CMD_DATA_STREAM_INTERVAL.
- rev. 0.16 – 10.08.2016: added MavLink parameters in CMD_READ_PARAMS_EXT2.
- rev. 0.17 – 21.10.2016: new commands CMD_CONTROL_CONFIG, CMD_CALIB_ORIENT_CORR; extended CMD_READ_PARAMS_EXT2.
- rev. 0.18 – 23.03.2017: new parameter FRAME_HEADING in CMD_HELPER_DATA; new flag CONTROL_FLAG_AUTO_TASK in CMD_CONTROL; new command CMD_CALIB_ACC_EXT_REF; document structure is updated.
- rev. 0.19 – 08.09.2017: add CMD_EVENT; updated CMD_DATA_STREAM_INTERVAL.
- rev. 0.20 – 10.30.2017: updated parameter EXT_FC_GAIN in CMD_READ_PARAMS_3; updated CMD_READ_PARAMS_EXT2.
- rev. 0.21 – 10.01.2018: new parameters in CMD_READ_PARAMS_EXT2;
- rev. 0.24 – 29.03.2018: updated CMD_EXECUTE_MENU; add AHRS_DEBUG_INFO and MOTOR4_CONTROL data structures; extended CMD_REALTIME_DATA_CUSTOM; add CMD_EXT_IMU_DEBUG_INFO; add CMD_READ_PARAMS_EXT3, CMD_WRITE_PARAMS_EXT3;
- rev. 0.25 – 27.11.2018: add protocol V2 specification.
- rev. 0.26 - 21.03.2019: add command CMD_AUTO_PID2.
- rev. 0.27 – 18.09.2019: add command CMD_EXT_IMU_CMD.
- rev. 0.28 – 31.01.2020: extended CMD_READ_PARAMS_EXT3.
- rev. 0.29 – 01.04.2020: add FLAGS description in CMD_RESET; updated CMD_AHRS_HELPER (extended examples section and described additional flags).
- rev. 0.30 – 10.09.2020: add CMD_READ_STATE_VARS; new flag in CMD_AHRS_HELPER; added Appendix D.
- rev. 0.31 – 10.11.2019: fixed CMD_READ_PARAMS_EXT;
- rev. 0.32 – 11.10.2021: added command description: CMD_CALIB_COGGING, CMD_CAN_DEVICE_SCAN, CMD_WRITE_PARAMS_SET
- rev. 0.33 - 02.11.2022: added CMD_EXT_SENS_CMD;
- rev. 0.40 – 13.02.2022: document structure was redesigned and added table of contents
- rev. 0.41 – 15.05.2024: added CAN_DRV_TELEMETRY; EXT_MOTORS_xx commands
- rev. 0.42 – 27.05.2024: added CMD_CONTROL_QUAT_x commands

Overview

Serial API allows for an external application or device to communicate with the SimpleBGC controller via serial port (UART). Each controller has several UART ports that can be used to send or receive Serial API commands. All models are equipped with the USB port that is visible as Virtual Com Port (VCP) for the host machine. Depending on controller, USB may be dedicated or shared with the one of UARTs.

Commands may be used to retrieve actual system state and realtime data, change settings, control gimbal, trigger pin state, execute various actions, get access to internal EEPROM and I2C bus, and so on. Moreover, SimpleBGC GUI software uses the same Serial API protocol to communicate with the board, so all of its functions may be implemented in third-party applications.

Communications is initiated from the remote side (host) by sending *outgoing* commands. The controller may do some action and send response (for the host it is an *incoming* command).

Board can work on different serial baud rates, adjustable by the parameters, with the 115200 as default value. Host can automatically find the proper baud rate by sending the CMD_BOARD_INFO command in a loop, altering the speed ant waiting for a response, until valid response is received, or should allow to specify a baud rate in its settings.

Additionally, board can auto-detect the "parity" parameter. EVEN and NONE parity are supported (NONE is selected by default after start, and EVEN is detected automatically). It means that beside the baud rates, host application should vary the parity setting, when connecting through an intermediate layer that can have this parameter unknown (like Bluetooth modules). For the direct UART or USB VCP connection, it is enough to set parity to "NONE".

Throughout capacity

The controller parses incoming command queue each 8ms, so there is no reason to send commands of the same type with the higher rate. Commands of different type may be sent without delay between them. It is responsibility of the host application to prevent an overflow of the input and output buffers of the controller (255 bytes each). If new serial data comes when the input buffer is full, the whole message will be lost. If controller has to generate an answer that does not fit into the output buffer, it hangs until buffer will have enough space to accept new data. It may negatively affect the normal operation and even make whole system unstable. The only exception is the CMD_REALTIME_DATA_xx and several others, that are considered as non-obligatory for delivery.

You can calculate the safe rates according to the size of incoming and outgoing commands and the configured baud rate for the serial port. Take into account the bandwidth and the buffering strategy of the intermediate transmission layer. For example, BLE modules have a very limited bandwidth and small buffers. Also, almost all radio modems have effective transmission rate less then 100Kbit/s in optimal conditions.

Debugging

You can configure SimpleBGC32 GUI to display all incoming and outgoing commands that it receives/sends. To do it, run it in a "console" mode using the "run_console.bat" script. Commands will be displayed in the "Debug" tab in format:

```
<local_time> <direction>: [<command_id>,<payload_length>] <payload_data_hex>
```

Note that several commands with high rate are not displayed (like CMD_REALTIME_DATA_xx).

Starting from the firmware and GUI version 2.66b4, it's possible to monitor all serial API messages on all other ports, by connecting GUI to any available serial port, configured for the SBGC Serial API mode, and enabling the "Debug" – "Set as debug port" option for it. Controller will forward all incoming and outgoing Serial API commands from all other ports to this port. Commands will be displayed in the "Debug" tab in format:

```
<MCU_time> port<idx>.<direction>: [<command_id>,<payload_length>] <payload_data_hex>
```

In this case, the "in" direction means "to the board", "out" – "from the board".
Note, that only the successfully parsed commands are forwarded. All unknown data is ignored.

Message format

Each command consists of the *header* and the *body*, both with checksum. Commands with the wrong header or body checksum, or with the body size that differs from expected, should be ignored. Parser should scan incoming datastream for the next start character and try to restore synchronization from it.

Input and output commands have the same format.

Protocol version 1

	header			body			
start character > (0x3E)	command ID, 0..255	payload size N=0..255	header checksum	payload variable size N			payload checksum
0	1	2	3	4	...	4+N-1	4+N

Header checksum is calculated as (command ID + payload_size) modulo 256.
Payload checksum is calculated as the sum of all payload bytes modulo 256.

Operation "modulo" means least significant byte of the sum.

Example: outgoing command to read Profile2:

	header			body	
0	1	2	3	4	5
0x3E	0x52	0x01	0x53	0x01	0x01

Protocol version 2

Starting from firmware version 2.68b0, firmware additionally supports protocol version 2, that has better error rejection by replacing the old 8 bit simple checksum (over payload only) to CRC16 checksum (over header + payload).

	header			payload			crc	
start character $ (0x24)	command ID, 0..255	payload size N=0..255	header checksum	variable size N			CRC16 (header + payload)	
0	1	2	3	4	...	4+N-1	4+N	4+N+1

Compared to version 1, it has a different start character "$" and a different checksum calculation:
payload checksum is calculated as a CRC16 over the header bytes and payload bytes, starting from index 1 to index 4+N-1. A reference implementation of CRC16 using polynomial 0x8005 is given in the Appendix A.

Protocol version 2 locking

At startup, firmware supports messages in both versions 1 and 2. But when the first valid message version 2 is received, this version is locked and all incoming messages in version 1 are not recognized anymore.

Data type notation

- 1u – 1 byte unsigned
- 1s – 1 byte signed
- 2u – 2 byte unsigned (little-endian order)
- 2s – 2 byte signed (little-endian order)
- 4f – float (IEEE-754 standard)
- 4s – 4 bytes signed (little-endian order)
- string – ASCII character array, first byte is array size
- Nb – byte array size N

Many parameters are grouped in arrays, that is indicated by the square brackets notation: "ANGLE[3]". Parameters that are split into axes, always go in the order ROLL, PITCH, YAW for the Euler angles and corresponding motors in a normal position. For the vectors order is X, Y, Z in the coordinate system having X pointing right, Y – forward, Z – up.

> **NOTE:** order of parameters in arrays is always ROLL,PITCH,YAW, but it doesn't corresponds to the order of Euler angles used to express a rotation – it is variable and defined by the parameter 'EULER_ORDER'.

Table of contents

Device information

Requests

CMD_BOARD_INFO (#86) – request board and firmware information

Simple format: no parameters

Extended format:

Name	Type	Min	Max	Possible values, remarks
CFG	2b			configuration for this serial driver: • for UARTs – period (in ms) between 20-bytes packets for BLE mode • for USB – not used
RESERVED	?			size is not checked

CMD_BOARD_INFO_3 (#20) – request additional board information

No parameters

Responses

CMD_BOARD_INFO (#86) – version and board information

Name	Type	Min	Max	Possible values, remarks
BOARD_VER	1u			Multiplied by 10: 3.0 => 30

FIRMWARE_VER	2u			Split into decimal digits X.XX.X, for example 2305 means 2.30b5
				`major_ver = (int)(FIRMWARE_VER/1000);`
				`minor_ver = (int)((FIRMWARE_VER%1000)/10);`
				`beta_ver = FIRMWARE_VER%10;`

STATE_FLAGS	1u			`bit0: DEBUG_MODE` – internal use only *Starting from frw.ver. 2.66:* `bit1: IS_FRAME_INVERTED` – system is re-configured for frame inversion over the middle motor; *The following flags are updated at the system initialization:* `bit2: INIT_STEP1_DONE` – finished initialization of all basic sensors, frame inversion configuration is applied; `bit3: INIT_STEP2_DONE` – finished initialization of the RC subsystem, adjustable variables, etc. Automated positioning is started; `bit4: STARTUP_AUTO_ROUTINE_DONE` - positioning and calibrations at startup is finished;

				`bit5: POWER_MANAGER_ENABLED` (for CAN_MCU board only) *Note: the extended version of system state flags is available in other commands, see SYSTEM_STATE_FLAGS*

BOARD_FEATURES	2u			`Bit set to encode functions firmware supports:`
				`BAT_MONITORING = (1<<1)`
				`3AXIS = (1<<0)`
				`ENCODERS = (1<<2)`
				`BODE_TEST = (1<<3)`
				`SCRIPTING = (1<<4)`
				`CURRENT_SENSOR = (1<<5)`
				`MAG_SENSOR = (1<<6)`
				`ORDER_OF_AXES_LETUS = (1<<7)`
				`IMU_EEPROM = (1<<8)`
				`FRAME_IMU_EEPROM = (1<<9)`
				`CAN_PORT = (1<<10)`
				`MOMENTUM = (1<<11)`
				`COGGING_CORRECTION = (1<<12)`
				`MOTOR4_CONTROL = (1<<13)`
				`ACC_AUTO_CALIB = (1<<14)`
				`BIG_FLASH = (1<<15)` – firmware needs 256Kb of FLASH

CONNECTION_FLAG	1u			`Bit set:` `CONNECTION_USB = 1`
FRW_EXTRA_ID	4u			Used for specific builds only

BOARD_FEATURES_EXT	2u			`Extends the set of BOARD_FEATURES to 16..31 bit:`
				`EXT_IMU = (1<<16)`
				`STATE_VARS = (1<<18)`
				`POWER_MANAGEMENT = (1<<19)`
				`GYRO_ADVANCED_CALIB = (1<<20)`
				`LIMITED_VERSION = (1<<21)`
				`REACTION = (1<<22)`
				`ENCODER_LUT = (1<<23)`

RESERVED	3b			
BASE_FRW_VER	2u			For “experimental” versions encodes the main version which they are based on

CMD_BOARD_INFO_3 (#20) – additional board information

Name	Type	Min	Max	Possible values, remarks
DEVICE_ID	9b			Unique Id used to identify each controller in licensing system
MCU_ID	12b			MCU ID, unique
EEPROM_SIZE	4u			Size of available EEPROM in current device. Generally 32K bytes
SCRIPT_SLOT1_SIZE ... SCRIPT_SLOT5_SIZE	2u*5			size of user-written scripts stored in five slots, 0 if slot is empty.
PROFILE_SET_SLOTS	1u			bit0..bit5: bit is set if the corresponding profile set is not empty. bit0 for profile set#1, bit2 for profile set#2, bit5 for profile set backup
PROFILE_SET_CUR	1u	1	6	A number of currently selected profile set

FLASH_SIZE	1u			Actual FLASH memory size (a number of 32 Kb pages)
IMU_CALIB_INFO	2b			
SCRIPT_SLOT6_SIZE ... SCRIPT_SLOT10_SIZE	2u*5			size of user-written scripts stored in five additional slots, 0 if slot is empty.
HW_FLAGS	2u			Board hardware configuration Bit0: one-wire crypto IC is installed
RESERVED	17b			

Configuring gimbal

Requests

CMD_READ_PARAMS (#82),
CMD_READ_PARAMS_3 (#21) – request parameters from the board
CMD_READ_PARAMS_EXT (#33) – request extended parameters part1
CMD_READ_PARAMS_EXT2 (#62) – request extended parameters part2
CMD_READ_PARAMS_EXT3 (#104) – request extended parameters part3

(frw.ver. 2.66+)

Name	Type	Min	Max	Possible values, remarks
PROFILE_ID	1u	0	4	profile ID to load. If value >4, currently selected profile is loaded.

CMD_WRITE_PARAMS (#87),
CMD_WRITE_PARAMS_3 (#22) - write parameters to board and saves to EEPROM
CMD_WRITE_PARAMS_EXT (#34) – write extended parameters part1
CMD_WRITE_PARAMS_EXT2 (#63) – write extended parameters part2
CMD_WRITE_PARAMS_EXT3 (#105) – write extended parameters part3

(frw.ver. 2.66+)

Data structure is the same as for the corresponding CMD_READ_PARAMS_xx incoming command.
On success, confirmation CMD_CONFIRM is sent in response and new values are applied. Some changes require system to be restarted, so full reset occurs in 1 second after this command, if there are no other CMD_WRITE_PARAMSxx have came in that time. It's recommended to send these commands in the "configuration" mode, activated by the CMD_WRITE_PARAMS_SET.

CMD_WRITE_PARAMS_SET (#119) – start or end of the writing parameters sequence

(frw.ver. 2.70b4+)

Send this command before sending the sequence of CMD_WRITE_PARAMSxx to enter the "configuration" mode, and send it again at the end of the sequence to apply changes and switch to a normal working mode. In the configuration state motors are turned OFF and system does not try to make initialization after each CMD_WRITE_PARAMSxx.

Name	Type	Min	Max	Possible values, remarks
ACTION	1u			1 – start writing parameters 0 – finish writing parameters

On success, confirmation CMD_CONFIRM is sent in response with the DATA=ACTION.

CMD_USE_DEFAULTS (#70) – reset to factory defaults

Name	Type	Min	Max	Possible values, remarks

PROFILE_ID	1u	0	4	profile ID to reset. Special values: 253 – erase EEPROM 254 – reset currently selected profile

CMD_CALIB_OFFSET (#79)– calibrate follow offset

No parameters

CMD_READ_PROFILE_NAMES (#28) – Request profile names stored in EEPROM

No parameters

CMD_WRITE_PROFILE_NAMES (#29) – Writes profile names to EEPROM

Name	Type	Min	Max	Possible values, remarks
PROFILE_NAME[5]	48b*5			Each name is encoded in UTF-8 format and padded with '\0' character to 48 byte size

CMD_PROFILE_SET (#95) – manage profile sets

(frw. ver. 2.65+)

Name	Type	Min	Max	Possible values, remarks
SLOT	1u	1	6	Slot to operate. 1..5: regular slots, 6 – backup slot
ACTION	1u			**PROFILE_SET_ACTION_SAVE = 1** save current configuration (including all profiles and simple calibrations) to the given slot **PROFILE_SET_ACTION_CLEAR = 2** cleat the selected slot **PROFILE_SET_ACTION_LOAD = 3** load configuration from the given slot
RESERVED	8b			

Confirmation is sent on success.

Responses

CMD_READ_PARAMS_3 (#21) – read/write system configuration part 1

Receive parameters for a single profile.

Name		Type	Min	Max	Possible values, remarks
PROFILE_ID		1u			profile ID to read or write. To access current (active) profile, specify 255. Possible values: 0..4
	P	1u	0	255	
	I	1u	0	255	divided by 100 when displayed in the GUI
	D	1u	0	255	

axis = (1..3)	POWER	1u	0	255	
	INVERT	1u	0	1	
	POLES	1u	0	255	
ACC_LIMITER_ALL		1u	0	255	*Units: 5 degrees/sec^2 0 – disabled.* (starting from ver. 2.60 is deprecated; replaced by the ACC_LIMITER3)
EXT_FC_GAIN[2]		1s*2	-127	127	
axis = (1..3)	RC_MIN_ANGLE	2s	-720	720	*Units: degrees*
	RC_MAX_ANGLE	2s	-720	720	*Units: degrees*

	RC_MODE	1u			`0..2 bits – mode:`
					`RC_MODE_ANGLE = 0`
					`RC_MODE_SPEED = 1`
					`3rd bit – control is inverted, if set to 1`

	RC_LPF	1u	0	15 (255)*	**Range depends on the flag "Extend LPF range" in GUI settings*
	RC_SPEED	1u	0	255	
	RC_FOLLOW	1u	-127	127	`ROLL, PITCH: this value specify follow rate for flight controller.` `YAW: if value != 0, "follow motor" mode is enabled.`
GYRO_TRUST		1u	0	255	
USE_MODEL		1u	0	1	

PWM_FREQ		1u			`PWM_FREQ_LOW = 0`
					`PWM_FREQ_HIGH = 1`
					`PWM_FREQ_ULTRA_HIGH = 2`

SERIAL_SPPED		1u			Baud rate for the main UART1 port (where USB normally connects)
					`115200 = 0`
					`57600 = 1`
					`38400 = 2`
					`19200 = 3`
					`9600 = 4`
					`256000 = 5`

RC_TRIM[3]		1s*3	-127	127	
RC_DEADBAND		1u	0	255	
RC_EXPO_RATE		1u	0	100	

RC_VIRT_MODE		1u			The mode of the RC_ROLL input pin operation:
					`RC_VIRT_MODE_NORMAL = 0`
					`RC_VIRT_MODE_CPPM = 1`
					`RC_VIRT_MODE_SBUS = 2`
					`RC_VIRT_MODE_SPEKTRUM = 3`
					`RC_VIRT_MODE_API = 10`

RC_MAP_ROLL RC_MAP_PITCH		1u*6			Assign input as a signal source. Bits 0..4 for channel number, bits 5..7 for a type. Value 0 means that input is not assigned.

RC_MAP_YAW
RC_MAP_CMD
RC_MAP_FC_ROLL
RC_MAP_FC_PITCH
				PWM source
				`RC_INPUT_ROLL = 1`
				`RC_INPUT_PITCH = 2`
				`EXT_FC_INPUT_ROLL = 3`
				`EXT_FC_INPUT_PITCH = 4`
				`RC_INPUT_YAW = 5`
				
				Analog source
				Channel = 1..3, type = 32 (5th bit is set)
				`ADC1 = 33`
				`ADC2 = 34`
				`ADC3 = 35`
				
				RC Serial source (CPPM/SBUS/SPEKTRUM):
				Virtual channel (1..31), type = 64 (6th bit is set)
				
				API Virtual control source
				Virtual channel (1..31), type = 128 (7th bit is set)
				
				Step signal source *(ver. 2.66+)*
				Step signal channel 1..6, type = 160 (5th and 7th bits are set)

RC_MIX_FC_ROLL
RC_MIX_FC_PITCH
	1u
	1u
				Mix the value received from the FC channel, to the value received from the selected RC channels, with the given rate:
				
				bits 0..5: mix rate. For example,
				`0 - no mix (100% RC)`
				`32 - 50% RC, 50% FC,`
				`63 - 0% RC, 100% FC`
				bits 6,7: target RC channel
				`0 - no mix`
				`1 - ROLL`
				`2 - PITCH`
				`3 - YAW`

FOLLOW_MODE	1u			`FOLLOW_MODE_DISABLED=0`
				`FOLLOW_MODE_FC=1`
				`FOLLOW_MODE_PITCH=2`

FOLLOW_DEADBAND	1u	0	255	
FOLLOW_EXPO_RATE	1u	0	100	
FOLLOW_OFFSET[3]	1s*3	-127	127	Starting from frw. ver. 2.70+ replaced by the FOLLOW_OFFSET_EXT[3]

AXIS_TOP
AXIS_RIGHT
FRAME_AXIS_TOP
FRAME_AXIS_RIGHT
	1s
	1s
	1s
	1s
				Main IMU and frame IMU orientation:
				`X = 1`
				`Y = 2`
				`Z = 3`
				`-X = -1`
				`-Y = -2`
				`-Z = -3`

FRAME_IMU_POS	1u			Location of the frame IMU:
				`FRAME_IMU_DISABLED = 0`
				`FRAME_IMU_BELOW_YAW = 1`
				`FRAME_IMU_ABOVE_YAW = 2`
				`FRAME_IMU_BELOW_YAW_PID_SOURCE = 3`

GYRO_DEADBAND	1u	0	255	*Units: 0.1 of gyro sensor's units.*
GYRO_SENS	1u			deprecated

I2C_SPEED_FAST	1u	0	1	If set, use 800kHz ultra-fast speed mode, otherwise use 400kHz speed

SKIP_GYRO_CALIB	1u			Skip calibration of gyroscope.
				`0 – do not skip`
				`1 – skip always`
				`2 – try to calibrate but skip if motion is detected`

RC_CMD_LOW RC_CMD_MID RC_CMD_HIGH MENU_BTN_CMD_1 MENU_BTN_CMD_2 MENU_BTN_CMD_3 MENU_BTN_CMD_4 MENU_BTN_CMD_5 MENU_BTN_CMD_LONG	1u*9			Assign action to various event sources. See CMD_EXECUTE_MENU for available actions

MOTOR_OUTPUT[3]	1u*3			Motor output mapping
				`DISABLED = 0`
				`ROLL = 1`
				`PITCH = 2`
				`YAW = 3`
				`I2C_DRV#1 = 4`
				`I2C_DRV#2 = 5`
				`I2C_DRV#3 = 6`
				`I2C_DRV#4 = 7`

BAT_THRESHOLD_ALARM	2s	-3000	3000	Negative means means alarm is disabled *Units: 0.01V*
BAT_THRESHOLD_MOTORS	2s	-3000	3000	Negative value means function is disabled *Units: 0.01V*
BAT_COMP_REF	2s	-3000	3000	Negative value means compensation is disabled. *Units: 0.01V*

BEEPER_MODES	1u			`BEEPER_MODE_CALIBRATE=1`
				`BEEPER_MODE_CONFIRM=2`
				`BEEPER_MODE_ERROR=4`
				`BEEPER_MODE_ALARM=8`
				
				`BEEP_BY_MOTORS=128`
				(if this flag is set, motors emit sound instead of internal buzzer)

FOLLOW_ROLL_MIX_START	1u	0	90	
FOLLOW_ROLL_MIX_RANGE	1u	0	90	
BOOSTER_POWER[3]	1u*3	0	255	Additional power to correct lost synchronization
FOLLOW_SPEED[3]	1u*3	0	255	
FRAME_ANGLE_FROM_MOTORS	1u	0	1	
RC_MEMORY[3]	2s*3	-36767	32767	Initial angle that is set at system start-up, in 14bit resolution *`Units: 0,02197265625 degree`*
SERVO1_OUT	1u*4			Disabled = 0

SERVO2_OUT SERVO3_OUT SERVO4_OUT				1..32 - Virtual channel number as source of data to be output
SERVO_RATE	1u	5	40	PWM frequency, 10 Hz per unit.

ADAPTIVE_PID_ENABLED	1u			`Set of bits (0 – disable all):`
				`EN_ROLL = 1`
				`EN_PITCH = 2`
				`EN_YAW = 4`

ADAPTIVE_PID_THRESHOLD	1u	0	255	
ADAPTIVE_PID_RATE	1u	1	255	
ADAPTIVE_PID_RECOVERY_FACTOR	1u	0	10	
FOLLOW_LPF[3]	1u*3	0	15	

GENERAL_FLAGS1	2u			`REMEMBER_LAST_USED_PROFILE = (1<<0)`
				`UPSIDE_DOWN_AUTO = (1<<1)`
				`SWAP_FRAME_MAIN_IMU = (1<<2)`
				`BLINK_PROFILE = (1<<3)`
				`EMERGENCY_STOP = (1<<4)`
				`MAGNETOMETER_POS_FRAME = (1<<5)`
				`FRAME_IMU_FF = (1<<6)`
				`OVERHEAT_STOP_MOTORS = (1<<7)`
				`CENTER_YAW_AT_STARTUP = (1<<8)`
				`SWAP_RC_SERIAL_UART_B = (1<<9)`
				`UART_B_SERIAL_API = (1<<10)`
				`BLINK_BAT_LEVEL = (1<<11)`
				`ADAPTIVE_GYRO_TRUST = (1<<12)`
				
				(frw. ver. 2.66+)
				`IS_UPSIDE_DOWN = (1<<13)`

PROFILE_FLAGS1	2u			`ADC1_AUTO_DETECTION = (1<<0)`
				`ADC2_AUTO_DETECTION = (1<<1)`
				`ADC3_AUTO_DETECTION = (1<<2)`
				`FOLLOW_USE_FRAME_IMU = (1<<4)`
				`BRIEFCASE_AUTO_DETECTION = (1<<5)`
				`UPSIDE_DOWN_AUTO_ROTATE = (1<<6)`
				`FOLLOW_LOCK_OFFSET_CORRECTION = (1<<7)`
				`START_NEUTRAL_POSITION = (1<<8)`
				`MENU_BUTTON_DISABLE_FOLLOW = (1<<9)`
				`TIMELAPSE_FRAME_FIXED = (1<<10)`
				`RC_KEEP_MIX_RATE = (1<<11)`
				`RC_KEEP_CUR_POS_ON_INIT = (1<<12)`
				(frw. ver. 2.66+)
				`OUTER_MOTOR_LIMIT_FREE_ROTATION = (1<<13)`
				(frw. ver. 2.69b3+)
				`GIMBAL_LOCK_SMOOTH_TRANSITION = (1<<14)`
				(frw. ver. 2.69b0+)
				`CAM_UPSIDE_DOWN_WORKING = (1<<15)`

SPEKTRUM_MODE	1u			`0 Auto-detection (default)`
				`1 DSM2/11ms/10bit`
				`2 DSM2/11ms/11bit`
				`3 DSM2/22ms/10bit`
				`4 DSM2/22ms/11bit`
				`5 DSMX/11ms/10bit`

				6 DSMX/11ms/11bit 7 DSMX/22ms/10bit 8 DSMX/22ms/11bit
ORDER_OF_AXES	1u			Order of hardware axes, counting from a camera: PITCH_ROLL_YAW = 0 YAW_ROLL_PITCH = 1 ROLL_YAW_PITCH* = 2 ROLL_PITCH_YAW = 3 * implemented in special builds of firmware only
EULER_ORDER	1u			Order of Euler angles to represent the current orientation of a camera and the target of stabilization: PITCH_ROLL_YAW = 0 ROLL_PITCH_YAW = 1 LOCAL_ROLL* = 2 ROLL_LOCAL* = 3 YAW_ROLL_PITCH = 4 YAW_PITCH_ROLL = 5 * used for 2-axis systems only
CUR_IMU	1u			currently selected IMU IMU_TYPE_MAIN=1 IMU_TYPE_FRAME=2
CUR_PROFILE_ID	1u			profile ID which is currently active in the controller, 0...4

CMD_READ_PARAMS_EXT (#33) – read/write system configuration part 2

Name		Type	Min	Max	Possible values, remarks
PROFILE_ID		1u			profile ID to read or write. To access current (active) profile, specify 255. Possible values: 0..4
N = (1..3)	NOTCH_FREQ[3]	1u*3	0	255	Center frequency, x2 Hz (value 10 means 20Hz), for each axis R,P,Y
	NOTCH_WIDTH[3]	1u*3	0	255	Width of -3dB gain band, Hz, for each axis R,P,Y
LPF_FREQ[3]		2u*3	0	1000	Low-pass filter -3dB cut-off frequency, Hz
FILTERS_EN[3]		1u*3			Set of bits (0 – disable all): EN_NOTCH1 = 1 EN_NOTCH2 = 2 EN_NOTCH3 = 4 EN_LPF = 8
ENCODER_OFFSET[3]		2s*3			*Units: 0,02197265625 degree*
ENCODER_FLD_OFFSET[3]		2s*3			*Units: 0,02197265625 degree*
ENCODER_MANUAL_SET_TIME[3]		1u*3	0	255	*Units: 10ms*
MOTOR_HEATING_FACTO		1u*3	0	255	

R[3]				
MOTOR_COOLING_FACTOR[3]	1u*3	0	255	
RESERVED	2b			
FOLLOW_INSIDE_DEADBAND	1u	0	255	
MOTOR_MAG_LINK[3]	1u*3	0	255	Deprecated, replaced by the MOTOR_MAG_LINK_FINE
MOTOR_GEARING[3]	2u*3			Real number encoded as 8.8 fixed point (1.0f → 256)
ENCODER_LIMIT_MIN[3] ENCODER_LIMIT_MAX[3]	1s*3 1s*3	-127	127	*Units: 3 degree* Startig from ver. 2.61 is deprecated, replaced by the FRAME_CAM_ANGLE_MIN.
NOTCH1_GAIN[3] NOTCH2_GAIN[3] NOTCH3_GAIN[3]	1s*3 1s*3 1s*3	-100	100	Notch gain, in dB (positive – notch, negative – peak filter)
BEEPER_VOLUME	1u	0	255	
ENCODER_GEAR_RATIO[3]	2u*3			*Units: 0.001*

ENCODER_TYPE[3]	1u*3			`Bits 0..3:`
				`ENC_TYPE_AS5048A = 1`
				`ENC_TYPE_AS5048B = 2`
				`ENC_TYPE_AS5048_PWM = 3`
				`ENC_TYPE_AMT203 = 4`
				`ENC_TYPE_MA3_10BIT = 5`
				`ENC_TYPE_MA3_12BIT = 6`
				`ENC_TYPE_ANALOG = 7`
				`ENC_TYPE_I2C_DRV1 = 8`
				`ENC_TYPE_I2C_DRV2 = 9`
				`ENC_TYPE_I2C_DRV3 = 10`
				`ENC_TYPE_I2C_DRV4 = 11`
				`ENC_TYPE_AS5600_PWM = 12`
				`ENC_TYPE_AS5600_I2C = 13`
				`ENC_TYPE_RLS_ORBIS = 14`
				`TYPE_RLS_ORBIS_PWM = 15`
				`Bit 4:`
				`SKIP_DETECTION = 1`
				`Bit 7:`
				`ENCODER_IS_GEARED = 1`

ENCODER_CFG[3]	1u*3			`For SPI encoders:`
				`SPI_SPEED_1MHz = 0`
				`SPI_SPEED_2MHz = 1`
				`SPI_SPEED_4MHz = 2`
				`SPI_SPEED_500kHz = 3`
				`For I2C_DRV:`
				`internal encoder type`

OUTER_P[3]	1u*3	0	255	
OUTER_I[3]	1u*3	0	255	

MAG_AXIS_TOP
MAG_AXIS_RIGHT
	1s
	1s
				`X = 1`
				`Y = 2`
				`Z = 3`
				`-X = -1`

				-Y = -2 -Z = -3
MAG_TRUST	1u	0	255	
MAG_DECLINATION	1s	-90	90	*Units: 1 degree*
ACC_LPF_FREQ	2u	0	1000	*Units: 0.01Hz*
D_TERM_LPF_FREQ[3]	1u*3	0	60	*Units: 10Hz*

CMD_READ_PARAMS_EXT2 (#62) – read/write system configuration part 3

Name		Type	Min	Max	Possible values, remarks
PROFILE_ID		1u			profile ID to read or write. To access current (active) profile, specify 255. Possible values: 0..4
channel = (1..2)	MAV_SRC	1u			Disabled=0 UART1=1 RC_SERIAL=2 UART2=3 USB VCP=4
	MAV_SYS_ID	1u	0	255	
	MAV_COMP_ID	1u	0	255	

	MAV_CFG_FLAGS	1u			FLAG_BAUD_MASK = ((1<<0) | (1<<1) | (1<<2)) // baud rate idx 0..5
					FLAG_PARITY_EVEN = (1<<3) // even parity
					FLAG_HEARTBEAT = (1<<4) // send heartbeat
					FLAG_DEBUG = (1<<5) // send debug to GUI
					FLAG_RC = (1<<6) // use RC values

	MAV_RESERVED	4b			
MOTOR_MAG_LINK_FINE[3]		2u*3	0	65000	*Units: 0.01*
ACC_LIMITER[3]		1u*3	0	200	*Units: 5 degrees/sec²*
PID_GAIN[3]		1u*3	0	255	pid_gain_float[axis] = 0.1 + PID_GAIN[axis]*0.02
FRAME_IMU_LPF_FREQ		1u	0	200	*Units: Hz*
AUTO_PID_CFG		1u			See 'CFG_FLAGS' in the CMD_AUTO_PID
AUTO_PID_GAIN		1u	0	255	See 'GAIN_VS_STABILITY' in the CMD_AUTO_PID
FRAME_CAM_ANGLE_MIN[3] FRAME_CAM_ANGLE_MAX[3]		2s*3 2s*3			Software limits for motor's angles *(frw. ver. 2.61+)* *Units: 1 degree*
GENERAL_FLAGS2		2u			*(frw. ver. 2.61+)* SEARCH_LIMIT_ROLL = (1<<0) SEARCH_LIMIT_PITCH = (1<<1) SEARCH_LIMIT_YAW = (1<<2)

				(frw. ver. 2.62b7+)
				`AUTO_CALIBRATE_MOMENTUM = (1<<3)`
				`USE_MOMENTUM_FEED_FORWARD = (1<<4)`
				`MOTORS_OFF_AT_STARTUP = (1<<5)`
				`FC_BELOW_OUTER = (1<<6)`
				
				(frw. ver. 2.66+)
				`DO_NOT_CHECK_ENCODER_LIMITS = (1<<7)`
				`AUTO_SAVE_BACKUP_SLOT = (1<<8)`
				`FC_BELOW_MIDDLE = (1<<9)`
				`Note: if both flags FC_BELOW_OUTER and FC_BELOW_MIDDLE are set, it means FC position on the camera platform`
				(frw. ver. 2.67b2+)
				`ENVIRONMENT_TEMP_UNKNOWN = (1<<10)`
				`LPF_EXTENDED_RANGE = (1<<11)`
				`SAVE_SYSTEM_STAT = (1<<12)`
				`FLAG2_DISABLE_ACC = (1<<13)`
				`FLAG2_DISABLE_POWER_MANAGER = (1<<14)`
				`ALLOW_FRAME_IMU_AS_MAIN = (1<<15)`

AUTO_SPEED	1u	1	255	*(frw. ver. 2.61+)* Speed used in automated tasks. The same range as for the RC_SPEED parameter
AUTO_ACC_LIMITER	1u	1	255	*(frw. ver. 2.61+)* Acceleration limiter used in automated tasks. The same range as for ACC_LIMITER parameter *Units: 5 degrees/sec²*
IMU_ORIENTATION_CORR[3]	2s*3			*(frw. ver. 2.61+)* The rotation angle of correction of main IMU sensor misalignment over its local X,Y,Z axis. *Units: 0.01 degrees*
TIMELAPSE_TIME	2u			*(frw. ver. 2.60+)* Time for the time-lapse motion sequence *Units: seconds*
EMERGENCY_STOP_RESTART_DELAY	2u			*Units: ms*
TIMELAPSE_ACC_PART	1u	0	250	*Units: 0.2%*
MOMENTUM[3]	2u*3			*(frw.ver. 2.62b7+)*
MOMENTUM_CALIB_STIMULUS[3]	1u*3	1	255	*(frw.ver. 2.62b7+)*
MOMENTUM_ELITPICITY[3]	1u*3	1	255	*(frw.ver. 2.62b7+)* *Units: 0.05*
FOLLOW_RANGE[3]	1u*3	1	180	*(frw.ver. 2.62b7+)* *Units: degrees*

STAB_AXIS[3]	1u*3			*(frw.ver. 2.62b7+)*
				Bits0..1: axis assigned for each motor:
				`0 – default`
				`1 – ROLL`
				`2 - PITCH`
				`3 – YAW`
				
				Bits2..4: enable automatic selection of best matching axis:
				`bit2: ROLL`
				`bit3: PITCH`

				`bit4: YAW`
OUTER_MOT_TILT_ANGLE	1s	-90	90	*Units: degrees*
The following parameters are applied for the firmware ver. 2.66+				
STARTUP_ACTION[4]	1u*4			bits 0..6: action, as listed in the CMD_EXECUTE_MENU.CMD_ID bit7: if set, menu button should be pressed
STARTUP_ACTION_SRC[2][4]	1u*8			Signal source, as listed in the RC_MAP_ROLL parameter
STARTUP_ACTION_THRESHOLD[2][4]	1s*8			Threshold for RC signal on a given source, multiplied by 10.

FORCE_POSITION_CFG[3]	1u*3			bits 0..2: snap angle, one of the 0, 45, 90, 180
				bits 4..7: flags:
				`FORCE_POSITION_FLAG_BUTTON_PRESS = (1<<4)`
				`FORCE_POSITION_FLAG_STARTUP = (1<<5)`
				`FORCE_POSITION_FLAG_IGNORE_LIMITS = (1<<6)`
				`FORCE_POSITION_FLAG_FINE_ADJUST = (1<<7)`

N=1..6 STEP_SIGNAL_SRC	1u			Signal source, as listed in the RC_MAP_ROLL parameter

N=1..6 STEP_SIGNAL_CFG	1u			bits 0..2: number of steps, one of the `[2, 3, 5, 10, 15, 25, 50, 100]`
				bit 3: if set, menu button should be pressed
				bit 5: if set, initial value is zero
				bits 6..7: mode
				`MODE_LEVEL_LOW = 0`
				`MODE_LEVEL_HIGH = 1`
				`MODE_LEVEL_LOW_HIGH = 2`

N=1..5 RC_CALIB_SRC	1u			Signal source to apply calibration, as listed in the RC_MAP_ROLL parameter

N=1..5 RC_CALIB_OFFSET	1s			Calibration is applied by the rule:
				val = val + RC_CALIB_OFFSET*`(RC_RANGE/2/128)`;
				if(val > 0) val = val * (80 + RC_CALIB_POS_SCALE) / 100;
				else val = val * (80 + RC_CALIB_NEG_SCALE) / 100;

N=1..5 RC_CALIB_NEG_SCALE	1u			
N=1..5 RC_CALIB_POS_SCALE	1u			
PARKING_POS_CFG	1u			ROLL: bit 0 – negative border, bit 1 – positive border PITCH: bit 2 – negative border, bit 3 – positive border YAW: bit 4 – negative border, bit 5 – positive border
EXT_LED_PIN_ID	1u			Use this pin to duplicate the on-board LED function. Values are listed in the CMD_TRIGGER_PIN.PIN_ID
INTERRUPT_CFG	2u			bits 0..4: pin ID as listed in the CMD_TRIGGER_PIN.PIN_ID bit 5: generate interrupt on emergency stop bit 6: generate interrupt on entering parking position
OVERLOAD_TIME	1u			*Units: 100ms*
AUTO_PID_MOMENTUM	1u	0	255	
JERK_SLOPE[3]	1u*3			*Units: 40ms*
MAV_CTRL_MODE	1u	0	2	0 – disabled 1 – ROLL and PITCH axes 2 – all axes

RC_SERIAL_SPEED UART2_SPEED	1u*2			See the SERIAL_SPEED parameter definition
MOTOR_RES[3]	1u*3	0	255	Motor resistance (one phase) *Units: 100 mOhm*
CURRENT_LIMIT	2u	0	65535	*Units: 10mA*
MIDDLE_MOT_TILT_ANGLE	1s	-90	90	*(frw. ver. 2.67+)* *Units: degrees*

CMD_READ_PARAMS_EXT3 (#104) – read/write system configuration part 3

(frw.ver. 2.66+)

Name	Type	Min	Max	Possible values, remarks
PROFILE_ID	1u			profile ID to read or write. To access current (active) profile, specify 255. Possible values: 0..4
RESERVED	21b			

EXT_IMU_TYPE	1u			`MavLink1 = 1`
				`MavLink2 = 2`
				`Vectornav VN200 = 3`
				`Inertialsense uAHRS = 4`

EXT_IMU_PORT	1u			`Disabled = 0`
				`UART1 = 1`
				`RC_SERIAL = 2`
				`UART2 = 3`
				`USB VCP = 4`

EXT_IMU_POSITION	1u			`BELOW_OUTER = 1`
				`ABOVE_OUTER = 2`
				`BELOW_MIDDLE = 8`
				`MAIN_IMU = 9`

EXT_IMU_ORIENTATION	1u			index in array [X, Y, Z, -X, -Y, -Z] bit0..2 for the TOP axis bit3..5 for the RIGHT axis
EXT_IMU_FLAGS	2u			EXT_IMU_FLAG_ACC_COMP_ONLY = 2 EXT_IMU_FLAG_REPLACE = 4 EXT_IMU_FLAG_Z = 8 EXT_IMU_FLAG_H = 16 EXT_IMU_FLAG_FRAME_UPSIDE_DOWN_UPDATE = 32 EXT_IMU_FLAG_AS_FRAME_IMU = 64 EXT_IMU_FLAG_GYRO_CORR = 128 *(frw.ver. 2.68b7+)*
EXT_IMU_ALIGN_CORRECTION[3]	2s*3			Rotation over X,Y,Z axes *Units: 0.001 degrees*
EXT_IMU_STARTUP_DELAY	1u			*Units: 50ms*
EXT_IMU_GYRO_CORR_RATE	1u			Strength of the gyroscope correction by ext. IMU in the “online calibration” algorithm.
EXT_IMU_RESERVED	4b			
SOFT_LIMIT_WIDTH[3]	1u*3	1	255	Width of the software limits defined by the FRAME_CAM_ANGLE_MIN, FRAME_CAM_ANGLE_MAX *Units: 0.1 degrees*
ADC_REPLACE_SRC[3]	1u*3			See RC_MAP_ROLL description for possible values

GLOCK_MID_MOT_POS_CORR_RATE	1u	0	255	
EXTRA_BTN_CFG[5]	5b			Extra buttons connected to controller's pins. Bits0..4: MCU pin source, see PIN_ID in CMD_TRIGGER_PIN Bit6: latching mode if set Bit7: invert action if set
POWER_CTRL_CFG	8b			1u: overcurrent_protection, units: 0.5A 1u: power_on_delay, units: 100ms 1u: power_off_delay, units: 100ms 1u: power_on_limiter, 0..255 4b: reserved
RESERVED	3b			
CAN_IMU_EXT_SENS_TYPE	1u			0 disabled 1 KVH 1725 2 KVH 1750 (ACC 2G) 3 KVH 1750 (ACC 10G) 4 KVH 1750 (ACC 30G) 5 KVH 1775 (ACC 10G) 6 KVH 1775 (ACC 25G) 7 KVH 1760 8 ADXRS453 9 ADIS16460 10 STIM210 11 STIM300 12 SCHA63X 64 Vectornav VN100/200 (UART) 65 Vectornav VN100/200 (SPI)
PROFILE_FLAGS2	2u			FOLLOW_PITCH_DISABLED = (1<<0) LOW_ANGLE_PRIOR_ROLL = (1<<1) LOW_ANGLE_PRIOR_PITCH = (1<<2) LOW_ANGLE_PRIOR_YAW = (1<<3) HEADING_TRIPOD_MODE = (1<<4)
RESERVED	3b			
GENERAL_FLAGS3	4u			ENC_LUT_EN_ROLL = (1<<0) ENC_LUT_EN_PITCH = (1<<1) ENC_LUT_EN_YAW = (1<<2) MAVLINK_YAW_ABSOLUTE = (1<<3)
FOLLOW_OFFSET_EXT[3]	2s*3	-16384	16384	*Frw. ver. 2.70+*: replaces old 8-bit FOLLOW_OFFSET[3] *Units: 0,02197265625 degree*
MOTOR_STARTUP_DELAY	2u			*Units: ms*
IMU_MODEL_MAIN IMU_MODEL_FRAME	1u 1u			
STAB_THRESHOLD_ANGLE[3]	1u*3	15	80	
EXT_BUZZER_PIN	1u			
ENC_LIMIT_RETURN_SPEED[3]	1u*3			
TRIPOD_MODE_AUTO_THREASHOLD	1u			
RC_DEADBAND_PITCH RC_DEADBAND_YAW RC_EXPO_RATE_PITCH RC_EXPO_RATE_YAW	1u 1u 1u 1u			If the flag PROFILE_FLAGS2.RC_EXPO_DEADBAND_SPLIT is set, use this values for PITCH and YAW axes; otherwise, common RC_DEADBAND and RC_EXPO_RATE variables are used for all axes.
PROFILE_FLAGS3	4u			

DEFAULT_PROFILE		1u	0	4	
RETRACTED_ANGLE[3]		2s*3			
SHAKE_GENERATOR_CFG (size 22 bytes)					
	AMPLITUDE[3]	1u*3	0	255	*Applied in logarithmic scale in order ROLL, TILT, PAN*
	BASE_FREQ	1u	0	255	base_freq_hz = 0.3 + BASE_FREQ * 0.05
	FREQ_RANGE	1u	0	100	freq_range_mult = 1.5 + FREQ_RANGE * 0.1
	PAUSE_PERIOD	1u	0	255	pause_period_ms = 500 + PAUSE_PERIOD * 20
	PAUSE_BALANCE	1u	0	100	
	PAUSE_ATTENUATION	1u	0	100	
	PAUSE_RANDOMNESS	1u	0	100	
	PAUSE_PHASE_VAR	1u	0	100	
	RESONANCE_GAIN[3]	1u*3	0	255	resonance_gain_db = RESONANCE_GAIN * 0.2
	RESONANCE_FREQ	1u	0	255	resonance_freq_hz = 0.3 + RESONANCE_FREQ * 0.1
	FREQ_SHIFT[3]	1s*3	-127	127	Mapped to logarithmic scale 1/3 … 3
	RESERVED	5b			
RESERVED		93b			

CMD_READ_PROFILE_NAMES (#28) – receive profile names from EEPROM

Name	Type	Min	Max	Possible values, remarks
PROFILE_NAME[5]	48b*5			Each name is encoded in UTF-8 format and padded with '\0' character to 48 byte size

Calibrating

Requests

CMD_CALIB_ACC (#65) – calibrate accelerometer
CMD_CALIB_GYRO (#103) – calibrate gyroscope
CMD_CALIB_MAG (#59) – calibrate magnetometer

Simple format: no parameters. Starts regular calibration of currently active IMU, selected by the CMD_SELECT_IMU_3 command.

Extended format:

Name	Type	Min	Max	Possible values, remarks
IMU_IDX	1u			(0 – currently active IMU, 1 – main IMU, 2 – frame IMU)
ACTION	1u			1 – do regular calibration 2 – reset all calibrations and restart 3 – do temperature calibration 4 – enable temp. calib. data, if present, and restart 5 – disable temp. calib. data (but keep in memory), and restart 6 – copy calibration from the sensor's EEPROM to the main EEPROM ("restore factory calibration" option) 7 – copy calibration from the main EEPROM to the sensor's EEPROM
TIME_MS	2u	0	65535	Time for gyroscope calibration, in milliseconds. If set to 0, default time is used (~4 seconds), which is good balance between precision and speed.
RESERVED	8b			

If all parameters are valid, confirmation is sent immediately on reception and in the end of calibration.

CMD_CALIB_EXT_GAIN (#71) – calibrate EXT_FC gains

No parameters

CMD_CALIB_POLES (#80) – calibrate poles and direction

No parameters

CMD_CALIB_BAT (#66) - calibrate internal voltage sensor

Name	Type	Min	Max	Possible values, remarks
ACTUAL_VOLTAGE	2u			*Units: 0.01V*

Confirmation is sent.

CMD_ENCODERS_CALIB_OFFSET_4 (#26) - calibrate offset of encoders

No parameters.

(frw. ver. 2.68b7+) optional parameter FOR_MOTOR (1u): value 0..2 to calibrate offset only for the given motor ROLL, PITCH or YAW. Value 255 – for all motors.

CMD_ENCODERS_CALIB_FLD_OFFSET_4 (#27) - start field offset calibration of encoders

All parameters are optional. Note the version of the firmware where they started to be supported.

Name	Type	Min	Max	Possible values, remarks
CALIB_ANGLE[3] *(optional, frw. ver. 2.62b6+)*	2s*3	1	-	Angle range to move during calibration. If omitted, default is 40°. *Units: 0,02197265625 degree.*
CALIB_SPEED[3] *(optional, frw.ver. 2.71b1+)*	2s*3	1	-	Speed of movement during the calibration. If omitted, default is 100. *Units: 0,06103701895 deg./sec.*
CALIB_FLAGS *(optional, frw.ver. 2.70b8+)*	2u			`FLAG_IGNORE_IMU_CHECK (1<<0)` – ignore IMU angle vs motor angle validity check `FLAG_IGNORE_ENCODER_CHECK (1<<1)` – ignore encoder angle vs motor angle validity check

CMD_CALIB_ORIENT_CORR (#91) – start the calibration of sensor misalignment correction *(frw. ver. 2.61+)*

Name	Type	Min	Max	Possible values, remarks
RESERVED	16b			

Confirmation is sent immediately. After calibration is finished, CMD_READ_PARAMS_EXT2 is sent with new values in the IMU_ORIENTATION_CORR[3].

CMD_CALIB_ACC_EXT_REF (#94) – refine the accelerometer calibration of the main IMU sensor

(frw. ver. 2.62b7+, encoders)

Use this command to refine the ACC calibration in the main IMU sensor by providing the reference ACC vector from the external well-calibrated IMU in the frame's coordinates. By using three encoders, gimbal controller is able to convert it to the main IMU's local coordinates, compare to measured ACC vector and use it to refine existing calibration: zero offset for two horizontal axes and scale factor for the vertical axis.

Name	Type	Min	Max	Possible values, remarks
ACC_REF[3]	2s*3			Reference ACC vector [X,Y,Z] in gimbal frame's coordinates (X-axis points right, Y-axis points forward, Z-axis points down relative to frame). *Units: 1g/512 ≈ 0,019160156 m/s²*
RESERVED	14b			

Conditions:

- One of the sensor's axis should be aligned to a gravity vector with the 20-degree tolerance
- Existing ACC calibration should be good enough

Possible usage scenario:

1. Rotate gimbal to a leveled position by the CMD_CONTROL and run this command – X,Y-axis offset will be refined
2. Tilt gimbal 90-degree down and run it again – Z-axis offset and Y-axis scale will be refined.
3. Return gimbal back to leveled position and run it again – Z-axis scale will be refined. This is enough to have correct ACC readings inside the working range ROLL=0, PITCH = [0..90].

Calibration takes about 0.5 seconds (controller averages multiple data samples to reduce noise). Confirmation is sent only if all conditions are satisfied.

CMD_CALIB_COGGING (#93) – starts the motor non-linearities calibration

Name		Type	Min	Max	Possible values, remarks
ACTION		1u			1 – Calibrate 2 – Delete calibration data
AXIS_TO_CALIBRATE		1u			Bit0: ROLL Bit1: PITCH Bit2: YAW
for axis = (1..3)	ANGLE	2u	20	360	Angle to move, in degrees
	SMOOTH	1u	0	100	Smooth the resulting curve, in %
	SPEED	1u			Speed of rotation, in relative units
	PERIOD	2u			Expected period of non-linearity curve, in degrees. Leave 0 for auto-detection.
	RESERVED	9b			
ITERATIONS_NUM		1u	2	-	
RESERVED		9b			

Command CMD_CONFIRM is sent in response with the DATA = 1 or 2 on success, 254 on error. Another command CMD_CONFIRM with the DATA = 255 is sent when calibration finishes.

CMD_SYNC_MOTORS (#123) – mechanically align motors working in parallel for a single axis

(frw. ver. 2.70b9+)

Name	Type	Min	Max	Possible values, remarks
AXIS	1u			0 ROLL 1 PITCH 2 YAW
POWER	1u	0	255	Amount of power to apply to motor's winding in synchronous mode
TIME_MS	2u	0	65535	Power is applied for the given time, then motors are turned OFF
ANGLE	2u			Angle to rotate. 0 to hold the current position.

Command CMD_CONFIRM is sent in response immediately and another command CMD_CONFIRM with the DATA = 1 is sent when the calibration finishes.

Real-time state monitoring and diagnostics

Requests

CMD_REALTIME_DATA_CUSTOM (#88) – request configurable realtime data

(frw. ver. 2.60+)

Name	Type	Min	Max	Possible values, remarks
FLAGS	4u			Each bit specify which data to include in response `bit0:` IMU_ANGLES[3] `bit1:` TARGET_ANGLES[3] `bit2:` TARGET_SPEED[3] `bit3:` FRAME_CAM_ANGLE[3] `bit4:` GYRO_DATA[3] `bit5:` RC_DATA[6] `bit6:` Z_VECTOR[3], H_VECTOR[3] `bit7:` RC_CHANNELS[18] `bit8:` ACC_DATA[3] `bit9:` MOTOR4_CONTROL data structure `bit10:` AHRS_DEBUG_INFO data structure `bit11:` ENCODER_RAW24[3] `bit12:` IMU_ANGLES_RAD[3] `bit13:` SCRIPT_VARS_FLOAT[10] `bit14:` SCRIPT_VARS_INT16[10] `bit15:` SYSTEM_POWER_STATE data structure `bit16:` FRAME_CAM_RATE[3] `bit17:` IMU_ANGLES_20[3] `bit18:` TARGET_ANGLES_20[3] `bit19:` COMM_ERRORS `bit20:` SYSTEM_STATE_FLAGS *A detailed description of the data structure is provided in the CMD_REALTIME_DATA_CUSTOM response specification*
RESERVED	6b			

CMD_REALTIME_DATA (#68), CMD_REALTIME_DATA_3 (#23) – request real-time data, response is CMD_REALTIME_DATA_3

No parameters

CMD_REALTIME_DATA_4 (#25) – request extended real-time data, response is CMD_REALTIME_DATA_4

No parameters

CMD_DATA_STREAM_INTERVAL (#85) – register or update *data stream* – a commands sent by the controller with the fixed rate without request

(frw. ver. 2.60+) **or based on events** *(2.65+)*

For each serial interface, only one unique combination of CMD_ID + CONFIG bytes may be registered. If the

data stream is already registered, it will be updated. To unregister it, specify INTERVAL_MS=0. The total number of data streams over all serial interfaces is limited to 10.
Take care of the serial bandwidth: if data flow exceeds bandwidth, particular messages may be skipped.
The interval is maintained with the +-1ms tolerance for the individual sample, but the averaged sample rate exactly matches to the specified.

Name	Type	Min	Max	Possible values, remarks
CMD_ID	1u			Command ID to be sent by this data stream. All supported commands are listed for the "CONFIG" parameter below.
INTERVAL_MS	2u			*SYNC_TO_DATA = 0:* Interval between messages, in milliseconds. Value 1 means each cycle (0.8ms) *SYNC_TO_DATA != 0:* Sample rate divider Set value = 0 to unregister this data stream identified by the [CMD_ID, CONFIG] bytes.
CONFIG	8b			Configuration specific to each command: **CMD_REALTIME_DATA_3** **CMD_REALTIME_DATA_4** no parameters **CMD_REALTIME_DATA_CUSTOM** • **flags** – 4u, see command specification. **CMD_AHRS_HELPER** • **imu_type** – 1u (0 – main IMU, 1 – frame IMU). **CMD_EVENT** *(ver. 2.65b7+)* • **event_id** – 1u - One of the EVENT_ID_xx, see the CMD_EVENT command specification • **event_type** – 1u - a bitwise combination of the EVENT_TYPE_xx flags, see the CMD_EVENT command specification **CMD_CAN_DRV_TELEMETRY** • **flags** – 4u, see command specification. • **drv_id** – 1u, driver ID, 0..6 **CMD_EXT_MOTORS_STATE** • **for_motors** – 1u • **data_set** - 4u
SYNC_TO_DATA *(frw.ver 2.70b1)*	1u			If set, message is sent immediately after the specified type of data is updated. The parameter INTERVAL_MS=N specifies the sample rate divider (message is sent on each N-th update event). Data types: `IMU_ATTITUDE = 1` – IMU attitude (Euler angles and DCM), updated each 8ms
RESERVED	9b			

If the data stream is successfully registered or updated, the CMD_CONFIRM is sent in answer.

For the command **CMD_EVENT**, the behavior is different. This message is sent only once when the event is

triggered, so the parameter INTERVAL_MS does not matter and should be set to any non-zero value. But it is still used for the "continuous" events like EVENT_TYPE_HOLD. The "event_type" parameter can be used to select which events to report.

Examples:

- Send CMD_REALTIME_DATA_4 with the rate 20Hz:
  `19 32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`
- Send CMD_REALTIME_DATA_CUSTOM (IMU angles + RC target angles) with the rate 10Hz:
  `58 64 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`
- Send CMD_EVENT when the mode button is pressed and released (once), and held (at 10Hz):
  `66 64 00 01 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`

CMD_READ_RC_INPUTS (#100) - read values for the selected RC inputs

Name	Type	Min	Max	Possible values, remarks
CFG_FLAGS	2u			bit0: try to initialize input, if it was not used by the controller and was not initialized.
RC_SRC[N]	1u*N			List of signal sources. Possible values are listed in the RC_MAP_ROLL parameter.

In response, CMD_READ_RC_INPUTS is returned with the values for the requested RC sources.

CMD_GET_ANGLES (#73), CMD_GET_ANGLES_EXT (#61) - Request information related to IMU angles and RC control state

No parameters.

CMD_SELECT_IMU_3 (#24) – Select which IMU to calibrate or send realtime data

(for commands that don't specify IMU type explicitly)

Name	Type	Min	Max	Possible values, remarks
IMU_TYPE	1u			`IMU_TYPE_MAIN=1` `IMU_TYPE_FRAME=2`

If the selected IMU is not connected, command is ignored.

CMD_DEBUG_VARS_INFO_3 (#253) – request information about debug variables

Name	Type	Min	Max	Possible values, remarks
START_IDX	1u			*Frw.ver. 2.72b0+, optional* Defines which index to start from. In case if full list doesn't fit in a single command, allows to get it in several commands

CMD_DEBUG_VARS_3 (#254) – request values of debug variables

Name	Type	Min	Max	Possible values, remarks

SELECTED_MASK[N]	4u*N			*Frw.ver. 2.72b0+, optional* If specified, asks to send only the selected variables. Each bit in the mask corresponds to the index of the variable as it goes in CMD_DEBUG_VARS_INFO_3, starting from 0. If the total number of variables exceeds 32, several masks should be used, where N = last_idx/32 + 1

CMD_CALIB_INFO (#49) – request information required for the "Calibration helper" dialog window

Name	Type	Min	Max	Possible values, remarks
IMU_TYPE	1u			1 – main IMU, 2 – frame IMU
RESERVED	11b			

On success, CMD_CALIB_INFO is sent in response.

CMD_READ_STATE_VARS (#111)– request reading system persistent state variables, cumulative statistics and maintenance data

(min. frw.ver. 2.68b7, "Extended" family only)

No parameters. CMD_READ_STATE_VARS message is sent in response.

CMD_WRITE_STATE_VARS (#112) – write system persistent state variables, cumulative statistics and maintenance data

Data structure is the same as in the CMD_READ_STATE_VARS.
CMD_CONFIRM is sent in response on success.

CMD_SET_DEBUG_PORT (#249) – use this port for debugging

Forward all incoming and outgoing commands in other serial API ports to the current port. Only one port in the system may be configured for this role.

Name	Type	Min	Max	Possible values, remarks
ACTION	1u			0 – Stop using this port for debugging 1 – Start using this port for debugging
CMD_FILTER *"plus" version only*	4u			Set the following bits to prevent sending heavy-duty commands to the debug port: 0: CMD_REALTIME_DATA_3, 1: CMD_REALTIME_DATA_4, 2: CMD_REALTIME_DATA_CUSTOM, 3: CMD_DEBUG_VARS_3, 4: CMD_MAVLINK_DEBUG, 5: CMD_GET_ANGLES, 6: CMD_GET_ANGLES_EXT, 7: CMD_BODE_TEST_DATA, 8: CMD_HELPER_DATA, 9: CMD_AHRS_HELPER, 10: CMD_GYRO_CORRECTION, 11: CMD_CONTROL, 12: CMD_SET_ADJ_VARS_VAL, 13: CMD_API_VIRT_CH_CONTROL, 14: CMD_API_VIRT_CH_HIGH_RES

RESERVED	11b			

Command CMD_CONFIRM is sent in response. All in and out commands from other ports that were passed the filter, are sent in CMD_SET_DEBUG_PORT incoming command. If there are no enough room in TX buffer, command will be dropped.

CMD_EXT_MOTORS_STATE (#131) – request real-time data for external auxiliary motors

(frw. ver. 2.73.x)

Use this command to query the real-time state and current values of parameters related to motion control.

Name	Type	Min	Max	Possible values, remarks
FOR_MOTORS	1u			Bits 0...6 select the motors need to be reported
DATA_SET	4u			Bit mask defines which data to be included in the report. See the response specification.

Multiple CMD_EXT_MOTORS_STATE are sent in response (one for each command) for the selected motors.

CMD_CONTROL_QUAT_STATUS (#141) – request real-time data related to gimbal control in quaternions

(frw. ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks
DATA_SET	4u			Bit mask defining which data to send in response. See DATA_SET field specification in the response message.

The command `CMD_CONTROL_QUAT_STATUS` is sent in response with the requested data.

Responses

CMD_REALTIME_DATA_CUSTOM (#88) – configurable realtime data

(frw. ver. 2.60+)

Name	Type	Min	Max	Possible values, remarks
TIMESTAMP_MS	2u			Timestamp in milliseconds
The set of varaibles below depends on requested data, see the CMD_REALTIME_DATA_CUSTOM request specifications				
IMU_ANGLES[3]	2s*3			Main IMU angles (Euler) *Units: 0,02197265625 degree.*
TARGET_ANGLES[3]	2s*3			Target angles that gimbal should keep (Euler) *Units: 0,02197265625 degree.*

	TARGET_SPEED[3]	2s*3			Target speed that gimbal should keep, over Euler axes *Units: 0,06103701895 degree/sec*
	FRAME_CAM_ANGLE[3]	2s*3			Relative angle of joints (motors) *Units: 0,02197265625 degree.*
	GYRO_DATA[3]	2s*3			Data from the gyroscope sensor with the calibrations applied. *Units: 0,06103701895 degree/sec.*
	RC_DATA[6]	2s*6			RC data in high resolution, assigned to the ROLL, PITCH, YAW, CMD, FC_ROLL, FC_PITCH inputs. *Units: normal range is -16384..16384, -32768 is for 'undefined' signal*
	Z_VECTOR[3] H_VECTOR[3]	4f*6	-1.0f	1.0f	IMU attitude in a form of rotation matrix (2 rows as gravity and heading vectors, 3rd row can be calculated as cross-product of them). See Appendix D: Coordinate system conversions
	RC_CHANNELS[18]	2s*18			All RC channels captured from s-bus, spektrum or Sum-PPM inputs. *Mapped to -16384..16384, -32768 is for 'undefined' signal*
	ACC_DATA[3]	2s*3			Data from the accelerometer sensor with the calibrations applied, expressed in END coordinate system, sign is inverted. *Units: 1/512 G*
	AHRS_DEBUG_INFO	26b			See the AHRS_DEBUG_INFO specification
	MOTOR4_CONTROL	8b			See the `MOTOR4_CONTROL` specification
	ENCODER_RAW24[3]	3b*3			Encoder raw angles in a high resolution (24bit per full turn), 3 bytes for each encoder in a sequence for ROLL, PITCH, YAW motors, lower byte first. Total 9 bytes. *(frw. ver. 2.68+)*
	IMU_ANGLES_RAD[3]	4f*3	-Pi	Pi	Main IMU Euler angles in radians *(frw. ver. 2.68b7+)*
	SCRIPT_VARS_FLOAT[10]	4f*10			Script variables in floats
	SCRIPT_VARS_INT16[10]	2s*10			Script variables in 16-byte signed integers
SYSTEM_POWER_STATE *(frw. ver. 2.70b6+) WARNING: specification is not final, may be changed in future!*					
motor =(1..3)	MOT_POWER	2s	-10000	10000	Effective power that produces torque, 10000 for 100% Encoder firmware: sign corresponds to the torque direction
motor =(1..3)	MOT_CURRENT	2u	0	65535	Estimated or measured current consumption per motor *Units: mA*
motor =(1..3)	MOT_TEMP	1s			Motor temperature estimated by heating model (if configured) or measured by the hardware sensors (if present) *Units: °C*
motor =(1..3)	MOT_FLAGS	2u			`bit0:` software limit in motor is violated `bit1:` current exceeds the limit `bit2:` motor driver is enabled (motor energized)
motor =(1..3)	MOT_RESERVED	6b			

	SYSTEM_POWER_STATE	1b			`POWER_STATE_ON_FROM_BACKUP = -2 // internal use`
					`POWER_STATE_STARTUP = -1 // internal use`
					`POWER_STATE_OFF = 0 // motors are OFF`
					`POWER_STATE_ON = 1 // motors are ON`
					`POWER_STATE_OFF_TEMPORARY = 2 // motors are temporarily OFF for calibration`
					`POWER_STATE_OFF_PARKING = 3 // motors are temporarily OFF when entering parking position`
					`POWER_STATE_ON_SAFE_STOP = 4 // motors are`

					`energized to softly drop the unbalanced payload before going OFF`
	BATTERY_VOLTAGE	2u			Voltage measured by the hardware voltage sensor *Units: 0.01V*
	TOTAL_CURRENT	2u	0	65535	Overall system current consumption measured by the hardware current sensor (if present) *Units: mA*
	SYSTEM_FLAGS	2u			`bit0:` software limit is violated in any motor `bit1:` overheat warning (estimated or measured temperature exceeds 80°C) `bit2:` internal driver OTW (over-temperature warning) signal `bit3:` internal driver FAULT signal
FRAME_CAM_RATE[3]		2s*3			Rate of rotation of frame-to-camera joints (motors) *(frw. ver. 2.70b6+)* *Units: 0,06103701895 degree/sec.*
IMU_ANGLES_20[3]		4s*3			Main IMU angles in 20bit resolution *(frw. ver. 2.70b8+)* *Units: 0,00034332275390625 degrees*
TARGET_ANGLES_20[3]		4s*3			Target angles in 20bit resolution *(frw. ver. 2.70b8+)* *Units: 0,00034332275390625 degrees*
COMM_ERRORS		frw.ver. 2.72b0			Communication errors
	I2C_ERRORS	2u			
	SERIAL_ERRORS	2u			
	CAN_ERRORS	2u			
	CAN_ERR_FLAGS	1u			bit0: err warn irq bit1: err passive irq bit2: bus off irq
SYSTEM_STATE_FLAGS *(frw. ver. 2.73.x)*		4u			`bit0..7` – see CMD_BOARD_INFO.STATE_FLAGS `bit8:` Shake Generator is enabled

CMD_REALTIME_DATA_3 (#23) - receive real-time data

Name		Type	Min	Max	Possible values, remarks
axis = (1..3)	ACC_DATA	2s			Data from the accelerometer sensor with the calibrations applied, expressed in END coordinate system, sign is inverted. *Units: 1/512 G*
	GYRO_DATA	2s			Data from the gyroscope sensor with the calibrations applied. *Units: 0,06103701895 degree/sec.*
SERIAL_ERR_CNT		2u	0	65535	

SYSTEM_ERROR		2u			Set of bits (0 – no error):
					`ERR_NO_SENSOR (1<<0)`
					`ERR_CALIB_ACC (1<<1)`
					`ERR_SET_POWER (1<<2)`
					`ERR_CALIB_POLES (1<<3)`
					`ERR_PROTECTION (1<<4)`
					`ERR_SERIAL (1<<5)`
					`Beside that, extended error contains bits:`
					`ERR_LOW_BAT1 (1<<6)`

				`ERR_LOW_BAT2 (1<<7)`
				`ERR_GUI_VERSION (1<<8)`
				`ERR_MISS_STEPS (1<<9)`
				`ERR_SYSTEM (1<<10)`
				`ERR_EMERGENCY_STOP (1<<11)`

SYSTEM_SUB_ERROR	1u			See Appendix E: "Emergency stop" error codes
RESERVED	3b			
RC_ROLL RC_PITCH RC_YAW	2s 2s 2s	1000	2000	RC control channels values (PWM or normalized analog)
RC_CMD	2s	1000	2000	RC command channel value (PWM or normalized analog)
EXT_FC_ROLL EXT_FC_PITCH	2s 2s	1000	2000	External FC PWM values. May be zero if their inputs are mapped to RC control or command.
IMU_ANGLE[3]	2s*3	-32768	32767	IMU angles in 14-bit resolution per full turn *Units: 0,02197265625 degree*
FRAME_IMU_ANGLE[3]	2s*3	-32768	32767	Angles measured by the second IMU (if present), in 14-bit resolution. *Units: 0,02197265625 degree*
TARGET_ANGLE[3]	2s*3	-32768	32767	Target angles, in 14-bit resolution *Units: 0,02197265625 degree*
CYCLE_TIME	2u			*Units: microseconds*
I2C_ERROR_COUNT	2u			Number of registered errors on I2C bus
ERROR_CODE	1u			deprecated, replaced by the SYSTEM_ERROR variable
BAT_LEVEL	2u			Battery voltage *Units: 0.01 volt*
RT_DATA_FLAGS	1u			bit0 set - motors are turned ON

CUR_IMU	1u			Currently selected IMU that provides angles and raw sensor data
				`IMU_TYPE_MAIN=1`
				`IMU_TYPE_FRAME=2`

CUR_PROFILE	1u	0	4	Currently selected profile
MOTOR_POWER[3]	1u*3	0	255	

CMD_REALTIME_DATA_4 (#25) - receive extended version of real-time data

Name	Type	Min	Max	Possible values, remarks
...The beginning of the message includes all data from the CMD_REALTIME_DATA_3				
FRAME_CAM_ANGLE[3]	2s*3			Relative angle of joints between two arms of gimbal structure, measured by encoder (with offset and gearing calibration is applied), by 2nd IMU or by other algorithms. Value 0 corresponds

				to normal position (each arms forms 90 degrees with the next order arm). *Units: 0,02197265625 degree*
RESERVED	1b			
BALANCE_ERROR[3]	2s*3	-512	512	Error in balance (0 – perfect balance, 512 - 100% of the motor power is required to hold a camera)
CURRENT	2u			Actual current consumption. *Units: mA*
MAG_DATA[3]	2s*3	-1000	1000	Raw data from magnetometer *Units: relative, calibrated for current environment to give ±1000 for each axis.*
IMU_TEMPERATURE FRAME_IMU_TEMPERATURE	1s 1s	-127	127	Temperature of IMU sensors. *Units: Celsius*
IMU_G_ERR	1u	0	255	Error between estimated gravity vector and reference vector for currently active IMU *Units: 0.1 degree*
IMU_H_ERR	1u	0	255	Error between estimated heading vector and reference vector for currently active IMU *Units: 0.1 degree*
MOTOR_OUT[3]	2s*3	-10000	10000	Motor effective output, proportional to torque. Max. value of ±10000 equals to applying full power. *(encoder firmware ver. 2.61+)*
CALIB_MODE	1u	0		If not 0, calibration or automatic task is performed: CALIB_MODE_EXT_GAIN 1 CALIB_MODE_SET_ANGLE_AND_SAVE 2 CALIB_MODE_POLES 3 CALIB_MODE_ACC 4 CALIB_MODE_GYRO 5 CALIB_MODE_ENCODER_OFFSET 6 CALIB_MODE_ENCODER_FLD_OFFSET 7 CALIB_MODE_AUTO_PID 8 CALIB_MODE_BODE_TEST 9 CALIB_MODE_GYRO_TEMP 10 CALIB_MODE_ACC_TEMP 11 CALIB_MODE_MAG 12 CALIB_MODE_SET_ANGLE 13 CALIB_MODE_SYSTEM_IDENT 14 CALIB_MODE_MOTOR_MAG_LINK 15 CALIB_MODE_SEARCH_LIMITS 16 CALIB_MODE_SET_OPERATION_POS 17 CALIB_MODE_IMU_ORIENTATION_CORR 18 CALIB_MODE_TIMELAPSE 19 CALIB_MODE_MOMENTUM 20 CALIB_MODE_MOMENTUM_AUTO 21 CALIB_MODE_COGGING 22 CALIB_MODE_ACC_EXT_REF 23 CALIB_MODE_SAFE_STOP 24 CALIB_MODE_ACC_SPHERE 25 CALIB_MODE_GYRO_AXES_ALIGNMENT 26 CALIB_MODE_EXT_IMU_GYRO 27 CALIB_MODE_EXT_IMU_ALIGN 28 CALIB_MODE_ACC_GYRO_MULTIPOINT 34
CAN_IMU_EXT_SENS_ERR	1u			Error code from the external sensor connected to the CAN_IMU (codes are specific to sensors).
ACTUAL_ANGLE[3]	2s*3			*Frw.ver. 2.72b0+* Depending on the current stabilization mode: - for inertial angles it's the same as `IMU_ANGLE[3]` - for motor-related modes (“Servo mode”, mixed inertial + motor

				having (M) notation), it encodes the motor angle
SYSTEM_STATE_FLAGS *(frw. ver. 2.73.x)*	4u			See CMD_REALTIME_DATA_CUSTOM.SYSTEM_STATE_FLAGS
RESERVED	18b			

CMD_GET_ANGLES (#73) - Information about actual gimbal control state

	Name	Type	Min	Max	Possible values, remarks
axis = (1..3)	IMU_ANGLE	2s			IMU angles in 14-bit resolution per full turn *Units: 0,02197265625 degree*
	TARGET_ANGLE	2s			Target angles, in 14-bit resolution *Units: 0,02197265625 degree*
	TARGET_SPEED	2s			Target speed that gimbal should keep, over Euler axes *Units: 0,1220740379 degree/sec*

CMD_GET_ANGLES_EXT (#61) - Information about angles in different format

	Name	Type	Min	Max	Possible values, remarks
axis = (1..3)	IMU_ANGLE	2s			IMU angles in 14-bit resolution per full turn *Units: 0,02197265625 degree*
	TARGET_ANGLE	2s			Target angles, in 14-bit resolution *Units: 0,02197265625 degree*
	FRAME_CAM_ANGLE	4s			Relative angle of joints between two arms of gimbal structure, measured by encoder or 2nd IMU. Value 0 corresponds to normal position of a gimbal. This angle does not overflow after multiple turns. *Units: 0,02197265625 degree*
	RESERVED	10b			

CMD_DEBUG_VARS_INFO_3 (#253) – receive a specification of the debug variables

	Name	Type	Min	Max	Possible values, remarks
DEBUG_VARS_NUM		1u	1	255	Total number of debug variables available. The following list may contain less number of variables, if it overflows the commands' limit. In this case, repeat the request CMD_DBUG_VARS_INFO_3 with the START_IDX = last_idx + 1.
var = (1...DEBUG_VARS_NUM)	VAR_NAME	string			1st byte is size, following by the ASCII characters. Note that '\0' character is not present at the end of the string.

	VAR_TYPE	1u			0..3bits - type:
					`VAR_TYPE_UINT8 = 1`
					`VAR_TYPE_INT8 = 2`
					`VAR_TYPE_UINT16 = 3`
					`VAR_TYPE_INT16 = 4`
					`VAR_TYPE_UINT32 = 5`
					`VAR_TYPE_INT32 = 6`
					`VAR_TYPE_FLOAT = 7 (IEEE-754)`
					
					4..7bits - flags:

					VAR_FLAG_ROLL = 16 its belong to ROLL axis VAR_FLAG_PITCH = 32 its belong to PITCH axis VAR_FLAG_YAW = 48 its belong to YAW axis VAR_FLAG_ANGLE14 = 64 its an angle (14bit per turn)
	RESERVED	2b			

CMD_DEBUG_VARS_3 (#254) – values of debug variables reflecting a state of the system.

The number of variables and their types are not strictly defined and may vary depending on the firmware version. Use CMD_DEBUG_VARS_INFO_3 to obtain a specification of the variables in run-time.

Name	Type	Min	Max	Possible values, remarks
SELECTED_MASK[N]	4u*N			*Frw.ver. 2.72b0, optional* This fields appears only if it was provided in the request; it copies the field from the request. See CMD_DEBUG_VARS_3 for details. If present, values are provided only for the selected variables with the index 'idx' obeying the rule: `(SELECTED_MASK[idx/32] & (1<<(idx%32))) != 0`
VAR_VALUE[N]	?			size and type of each variable is encoded by the CMD_DEBUG_VARS_INFO_3 structure

CMD_CALIB_INFO (#49) – receive information required for the "Calibration helper" dialog window.

Name	Type	Min	Max	Possible values, remarks
PROGRESS	1u	0	100	Progress of operation in percents
IMU_TYPE	1u			1 – main IMU, 2 – frame IMU
ACC_DATA[3]	2s*3			Data from the accelerometer sensor with the calibrations applied, expressed in END coordinate system, sign is inverted. *Units: 1/512 G*
GYRO_ABS_VAL	2u			Amplitude of gyro signal
ACC_CUR_AXIS	1u	0	2	ACC axis to be calibrated
ACC_LIMITS_INFO	1u			Bit set of calibrated limits, where bits 0...5 corresponds to the index in array [+X,-X,+Y,-Y,+Z,-Z]
IMU_TEMP_CELS	1s	-127	127	IMU temperature, Celsius
TEMP_CALIB_GYRO_ENABLED	1u	0	1	Set to 1 if gyro temperature calibration is enabled
TEMP_CALIB_GYRO_T_MIN_CELS TEMP_CALIB_GYRO_T_MAX_CELS	1s 1s	-127	127	Range of temperature calibration *Units: Celsius*
TEMP_CALIB_ACC_ENABLED				Set to 1 if ACC temperature calibration is enabled
TEMP_CALIB_ACC_SLOT_NUM[6]	1u*6	0	3	The number of calibrated temperature slots for accelerometer for each limit, in order [+X,+Y,+Z,-X,-Y,-Z]

TEMP_CALIB_ACC_T_MIN_CELS TEMP_CALIB_ACC_T_MAX_CELS	1s 1s			Range of temperature calibration *Units: Celsius*
H_ERR_LENGTH	1u	0	255	The length of error vector between estimated and referenced heading vectors. *Unit vector=100*
RESERVED	7b			

CMD_SCRIPT_DEBUG (#58) – state of execution of user-written script

Name	Type	Min	Max	Possible values, remarks
CMD_COUNT	2u			current command counter
ERR_CODE	1u			see error definitions in the CMD_WRITE_FILE command

CMD_ADJ_VARS_STATE (#46) – receive the state of adjustable variables

Name	Type	Min	Max	Possible values, remarks
Firmware ver. prior to 2.62b5				
TRIGGER_RC_DATA	2s	-500	500	RC signal for the "trigger" variable slot
TRIGGER_ACTION	1u	0	255	ID of the triggered action. The full set of actions is given in the specification of MENU_BTN_CMD_1..5 parameters
ANALOG_RC_DATA	2s	-500	500	RC signal for the "analog" variable slot
ANALOG_VALUE	4s			Current value of the variable after all calculations
RESERVED	6b			
Firmware ver. 2.62b5+				
TRIGGER_RC_DATA	2s	-16384	16384	RC signal for the "trigger" variable slot
TRIGGER_ACTION	1u	0	255	ID of the triggered action. The full set of actions is given in the specification of MENU_BTN_CMD_1..5 parameters
ANALOG_SRC_VALUE	2s	-16384	16384	Signal value requested in the ANALOG_SRC_ID
ANALOG_VAR_VALUE	4f			Value of variable requested in the ANALOG_VAR_ID
LUT_SRC_VALUE	2s	-16384	16384	Signal value requested in the LUT_SRC_ID. Always encoded in a range -16384..16384.
LUT_VAR_VALUE	4f			Current value of variable requested in the LUT_VAR_ID

CMD_READ_RC_INPUTS (#100) - answer to the requested RC sources

Name	Type	Min	Max	Possible values, remarks
RC_VAL[N]	2s*N	-16384	16384	Values for each RC source in order as requested in the incoming CMD_READ_RC_INPUTS command. A special value RC_UNDEF=-32768 returned if signal is absent.

CMD_READ_STATE_VARS (#111) – result of reading system persistent state variables, cumulative statistics and maintenance data

(min. frw.ver. 2.68b7, "Extended" family only)

Name	Type	Min	Max	Possible values, remarks
STEP_SIGNAL_VAL[6]	6*1u			Step signal current value
SUB_ERROR	1u			Last code of EMERGENCY_STOP error
MAX_ACC	1u			Max. registered acceleration *Units: 1/16G*
WORK_TIME	4u			Total working time *Units: seconds*
STARTUP_CNT	2u			Counter of system starts
MAX_CURRENT	2u			Max. registered instant current consumption *Units: mA*
IMU_TEMP_MIN IMU_TEMP_MAX	1u 1u			IMU temperature *Units: C*
MCU_TEMP_MIN MCU_TEMP_MAX	1u 1u			Main MCU temperature *Units: C*
SHOCK_CNT[4]	4*1u			shock detector counter for specified thresholds
ENEGRY_TIME	4u			Time collecting consumed energy statistics *Units: seconds*
ENERGY	1f			Total consumed energy *Units: Watt*hour*
AVG_CURRENT_TIME	4u			Time collecting average current statistics *Units: seconds*
AVG_CURRENT	1f			Average current *Units: A*
RESERVED	152b			Zero bytes to keep payload size exactly 192 bytes

CMD_SET_DEBUG_PORT (#249) – receive serial API commands from all other ports for a debugging

This is for debug mode initiated by the CMD_SET_DEBUG_PORT outgoing command.

Name	Type	Min	Max	Possible values, remarks
TIME_MS	2u			Time since system start, in milliseconds
PORT_AND_DIR	1u			bits0..6: port index bit7: direction
CMD_ID	1u			Command id
PAYLOAD	…			Payload, variable length

CMD_CAN_DRV_TELEMETRY (#127) – receive CAN_DRV telemetry

(min. frw.ver. 2.72b0, "Extended" family only)

Telemetry from CAN_DRV can be requested via command CMD_DATA_STREAM_INTERVAL

Name	Type	Min	Max	Possible values, remarks
DRV_ID	1u	0	6	Driver Id sending the telemetry
TIMESTAMP_MS	2u			System local time in milliseconds
DATA	?b			See below

The DATA field of this command depends on the 'flags' parameter of the request, where bits 0..31 defines the variables to be sent in response:

Bit	Name	Type	Possible values, remarks
0	CAN_BUS_FLAGS	1u	
1	CAN_BUS_ERR_CNT	2u	
2	TEMP_BOARD	1s	Units: °C
3	TEMP_MOTOR	1s	Units: °C
4	AVG_CURRENT	2u	Units: mA
5	BUS_VOLTAGE	2u	Units: mV
6	PIN_STATE	1u	State for some digital inputs, each bit is 0|1 for low/high state (see bits definition below)
7	PHASE_CURRENT[3]	2s*3	Units: mA
8	ENCODER_ERROR_CNT	2u	
9	IQ_CURRENT ID_CURRENT	2s 2s	Iq is effective current (providing torque) Units: mA
10	MOTOR_SPEED	2s	Units: RPM
11	PID_POS_ERR	2s	
12	PID_SPEED_ERR	2s	
13	IQ_REF	2s	Commanded torque, ±32768 corresponds to ±100%
14	PID_CURRENT_ERR	2s	

PIN_STATE bits:

Bit	Pin
0	LIMIT
1	INDEX
2	EMERGENCY
3	AUX1
4	AUX2
5	AUX3
6	HALF_BRIDGE_ENABLE
7	ENCODER_SPI_CS

CMD_EXT_MOTORS_STATE (#131) – real-time data related to auxiliary motors

(frw. ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks

MOTOR_ID	1u	0	6	Motor ID
DATA_SET	4u			Bit mask defining the data set. Copied from the request.
DATA	*			See specification below.

The DATA field of this command depends on the 'DATA_SET' parameter of the request, where bits 0..31 defines the variables to be included in response:

Bit	Name	Type	Possible values, remarks
0	CONTROL_MODE	1u	0: position 1: speed 2: torque
1	TORQUE	2s	±32767 for the maximum torque the driver can provide
2	TORQUE_SETPOINT	2s	
3	SPEED16	2s	*Units: 0,1220740379 deg./sec.*
4	SPEED16_SETPOINT	2s	
5	SPEED32	4s	*Units micro-radians/sec.*
6	SPEED32_SETPOINT	4s	
7	ANGLE16	2s	*Units: 0,02197265625 deg*
8	ANGLE16_SETPOINT	2s	
9	ANGLE32	4s	*Units: 0,00034332275390625 deg.*
10	ANGLE32_SETPOINT	4s	
11	STATE_FLAGS	2u	Bits 0..1: Motors ON/OFF state • 0: OFF BRAKE • 1: ON • 2: OFF FLOATING • 3: OFF SAFE STOP Bit2: INDEX_PIN_STATE TODO: • error flags
12	MAX_SPEED	2u	*Units equal to corresponding variables in the command CMD_EXT_MOTORS_CONFIGURE*
13	MAX_ACCELERATION	2u	
14	JERK_SLOPE	2u	
15	MAX_TORQUE	2u	
16	CURRENT	2u	*Units: mA*
17	BATTERY_VOLTAGE	2u	*Units: mV*
18	MOTOR_TEMPERATURE	1s	*Units: °C*
19	DRIVER_TEMPERATURE	1s	*Units: °C*

CMD_CONTROL_QUAT_STATUS (#141) – real-time data related to gimbal control in quaternions

(frw. ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks
DATA_SET	4u			Bit mask defining which data is included in this message.

DATA				See specification below.

DATA_SET field specification:

Bit	Name	Type	Possible values, remarks
0	MODE	1u	As specified in the last command CMD_CONTROL_QUAT
	FLAGS	2u	
1	TARGET_ATTITUDE[4]	4f*4	Externally commanded attitude as a target for a motion plan.
2	SETPOINT_ATTITUDE[4]	4f*4	Attitude currently set as a setpoint that gimbal stabilizes
3	ACTUAL_ATTITUDE[4]	4f*4	Attitude currently measured by the IMU
4	TARGET_SPEED[3]	2s*3	Externally commanded speed as a target for a motion plan
5	SETPOINT_SPEED[3]	2s*3	Current speed setpoint that gimbal stabilizes
6	ACTUAL_SPEED[3]	2s*3	Actual speed measured by the IMU gyroscope sensor
7	TARGET_ATTITUDE_PACKED	8b	A packed version of corresponding variables (see Appendix F for format description)
8	SETPOINT_ATTITUDE_PACKED		
9	ACTUAL_ATTITUDE_PACKED		

ATTITUDE in a form of unit quaternion [w, x, y, z] and SPEED in a form of rotation vector [x, y, z] with units 0,06103701895 degree/sec, both in END coordinate system (see Coordinate system conversions)

Run-time gimbal parameters

Requests

CMD_SET_ADJ_VARS_VAL (#31) – Update the value of selected parameter(s).

This command is intended to change parameters on-the-fly during system operation, and does not save parameters to EEPROM.
To save updated parameters permanently, use the CMD_SAVE_PARAMS_3 command.
The same command is also used as an outgoing command to read the value of adjustable variable(s).

Name		Type	Min	Max	Possible values, remarks
NUM_PARAMS		1u	1	40	Number of parameters in command
for N = (1..NUM_PARAMS)	PARAM<N>_ID	1u			ID of parameter. See the Appendix B: Run-time parameters definition (adjustable variables) for a list of available variables.
	PARAM<N>_VALUE ...	4b			Value depends on type of parameter. Values are packed according to C-language memory model, little-endian order. 1- or 2-byte types converted to 4-byte using C-language type conversions. Floats are packed according to IEEE-754.

On success, confirmation is sent in response.

CMD_GET_ADJ_VARS_VAL (#64) – Query the actual value of selected parameter(s).

This command requests actual values of adjustable parameters.
On success, CMD_SET_ADJ_VARS_VAL is sent in response.

Name		Type	Min	Max	Possible values, remarks
NUM_PARAMS		1u	1	40	Number of parameters in command
for N = (1..NUM_PARAMS)	PARAM<N>_ID	1u			ID of parameter. See the Appendix B: Run-time parameters definition (adjustable variables) for a list of available variables.

CMD_READ_ADJ_VARS_CFG (#43) – request configuration of mapping of control inputs to adjustable variables

CMD_READ_ADJ_VARS_CFG incoming command is sent in response.

CMD_WRITE_ADJ_VARS_CFG (#44) – writes configuration of mapping of control inputs to adjustable variables

Data format is the same as in corresponding CMD_READ_ADJ_VARS_CFG incoming command.
On success, confirmation is sent in response.

CMD_SAVE_PARAMS_3 (#32) – Saves current values of parameters linked to adjustable variables, to EEPROM

Use this command to save parameters updated by the "Adjustable Variables", permanently to EEPROM. For parameters that are split to profiles, only the current profile slot is updated.

Name	Type	Min	Max	Possible values, remarks
ADJ_VAR_ID_1 ADJ_VAR_ID_2 … ADJ_VAR_ID_N	1u*N			*frw.ver. 2.68b9+* Optional array of IDs of adjustable variables to save. If not specified, save all active adjustable variables.

CMD_ADJ_VARS_STATE (#46) – request the state of adjustable variable in the given trigger and analog slots.

Firmware ver. prior to 2.62b5:

Name	Type	Min	Max	Possible values, remarks
TRIGGER_SLOT	1u	0	9	
ANALOG_SLOT	1u	0	14	

Firmware ver. 2.62b5+:

Name	Type	Min	Max	Possible values, remarks
TRIGGER_SLOT	1u	0	9	"Trigger" slot number to show its state
ANALOG_SRC_ID	2u			Signal source to show its value
ANALOG_VAR_ID	1u			Variable ID to show its value
LUT_SRC_ID	2u			Signal source to show its value
LUT_VAR_ID	1u			Variable ID to show its value

Responses

CMD_SET_ADJ_VARS_VAL (#31) – receive the values of adjustable variables.

Sent as an answer on CMD_GET_ADJ_VARS_VAL. See corresponding outgoing command for format description: CMD_SET_ADJ_VARS_VAL

CMD_READ_ADJ_VARS_CFG (#43) – receive the configuration for adjustable variables

There are 10 “trigger” slots and 15 “analog” slots. “Trigger” type is used to execute action depending on the

RC signal level, where full range is split into 5 levels. “Analog” type is used to adjust parameter by RC signal. MIN_VAL and MAX_VAL specify a working range, that is mapped to a native range of particular parameter.

Name		Type	Min	Max	Possible values, remarks
slot = (1..10)	TRIGGER_SRC_CH	1u			See the *RC_MAP_ROLL* parameter definition
	TRIGGER_ACTION[5]	1u*5			See the *CMD_EXECUTE_MENU* command for a list of available actions
slot = (1..15)	ANALOG_SRC_CH	1u			See the *RC_MAP_ROLL* parameter definition
	VAR_ID	1u			bits0..6: the ID of variable. Full list of adjustable variables is given in the Appendix B bit7: if set, the value is processed as a "multiplier" for a given variable. *(frw. ver. 2.62b6+)*
	MIN_VAL	1u			
	MAX_VAL	1u			
RESERVED		8b			

IMU correction and diagnostic

Requests

CMD_HELPER_DATA (#72) – provide helper data for AHRS system

Use this command to increase precision of attitude estimation under certain conditions like curved or accelerated motion. More information in the Appendix C: Providing external reference attitude/heading information from UAV

Legacy format (prior to frw. ver. 2.60):

Name	Type	Min	Max	Possible values, remarks
FRAME_ACC[3]	2s*3	-	-	Linear acceleration of the frame, [X,Y,Z] components in a coordinate system **`COORD_SYS_GROUND_YAW_ROTATED`** (see description below). *Units: 1g/512 ≈ 0,019160156 m/s²*
FRAME_ANGLE_ROLL FRAME_ANGLE_PITCH	2s 2s	-32768	32767	Inclination of the outer frame in a given coordinate system. Pass zero values to not use this information. *Units: 0,02197265625 degree.*

Extended format (frw. ver. 2.60+):

Name	Type	Min	Max	Possible values, remarks
FRAME_ACC[3]	2s*3	-	-	Linear acceleration of the frame with the inverted sign. Vector with the [X,Y,Z] components in a given coordinate system (see FLAGS below). *Units: 1g/512 ≈ 0,019160156 m/s²*
FRAME_ANGLE_ROLL FRAME_ANGLE_PITCH	2s 2s	-32768	32767	Inclination of the outer frame in the `COORD_SYS_GROUND_YAW_ROTATED`. These angles are used only in encoders or 2nd IMU are not installed to roughly estimate the motor angles. Pass zero values to not use this information. *Units: 0,02197265625 degree.*
FLAGS	1u			**bits 0..2**: coordinate system where FRAME_ACC and FRAME_SPEED vectors are defined. **`COORD_SYS_GROUND_YAW_ROTATED = 1`** Global system rotated with the camera over Z axis: Y-axis is aligned with the main IMU's Y-axis (points forward), X-axis points right, Z-axis points down (nadir) **`COORD_SYS_GROUND = 2`** END Global system: X-axis matches true East ,Y-axis matches true North, Z-axis matches nadir. ***Notes:*** *END system differs from commonly used NED system. To convert, swap X and Y values in vectors.* *A magnetometer sensor should be installed and calibrated to give global reference for the main IMU. If no magnetometer present, Y-axis points arbitrary direction, so it is required to additionally provide the FRAME_HEADING parameter and use encoders to allow synchronization of the local coordinate system to earth-related system.*

				`COORD_SYS_FRAME = 3`
				Coordinate system linked to the gimbal's outer frame: Y-axis matches frame's "forward", X-axis matches frame's "right", Z-axis matches frame's "down". For example, having an accelerometer-measured vector `acc_raw` in NED system in units m/s/s, compensate it for the gravity `acc = acc_raw - acc_1g` and convert it the following way:
				`FRAME_ACC[0] = -acc.y*52.19164`
				`FRAME_ACC[1] = -acc.x*52.19164`
				`FRAME_ACC[2] = -acc.z*52.19164`
				
				Note: *one of the following conditions should be satisfied:*
				- a 2nd frame-mounted IMU and YAW encoder in the regular firmware
				
				bits 3..5: reserved
				
				bit6: if set, the FRAME_HEADING is assumed to be computed in Euler order "ROLL-PITCH-YAW" rather then default "PITCH-ROLL-YAW"
				(frw. ver. 2.70b7+)
				
				bit7: Use FRAME_HEADING parameter as a heading reference to align the IMU's local coordinate system to earth-related system, or to compensate gyro drift by the YAW axis if frame is fixed. If bit is not set, FRAME_HEADING is ignored
				(frw. ver. 2.62b7+)

FRAME_SPEED[3]	2s*3	-	-	Angular speed of the frame, [X,Y,Z] components in a given coordinate system. Helps to increase a precision of stabilization in systems w/out encoders or 2nd IMU. Pass zero values to not use this information. *Units: 0,06103701895 degree/sec*
FRAME_HEADING (frw. ver. 2.62b7+)	2s	-16384	16384	Angle of the frame relative to the North by the YAW axis. On first occurrence, YAW angle will be updated, taking into account the position of the main IMU relative to a frame. Then it will be used only as a reference for a gyro drift correction. If frame is fixed, it's enough to set this value once. But if frame is moving, it should be measured and update with the high enough rate (10-50Hz) to reflect the frame rotation. Remarks: *bit7 in the FLAGS parameter should be set to use this value. *Provided angle may be wrapped to +-180 degrees or 0..360 degrees. *Special value of 32767 stops the use of this reference and makes IMU heading unreferenced. *"Heading angle" is YAW angle expressed in Euler order PITCH-ROLL-YAW. Starting from firmware 2.70b7+, the FLAGS : bit6 can change it to "ROLL-PITCH-YAW". * This correction has a priority compared to the correction received from the external IMU, if it's connected. *Units: 0,02197265625 degree.*
RESERVED	1b			

For the lateral acceleration compensation, it is enough to provide only the FRAME_ACC data, leaving all other fields empty. Feed fresh ACC and angles data with the pretty low rate 10-20 Hz, because strong low-pass filter is applied internally. If the FRAME_SPEED data need to be provided, data rate should be much higher, up to 125 Hz.

How to ensure that the ACC correction is applied properly, on the bench:

1. Temporarily set the "ACC LPF" filter parameter in the GUI to 5-10Hz – it will remove noise but keep fast reaction of the "IMU_G_ERR" variable in the "Monitoring" tab of the GUI. This variable shows the distance between the estimated gravity vector and vector, measured by accelerometer.
2. Without motion, when you tilt the frame, the FRAME_ACC vector should have all components close to zero. The IMU_G_ERR variable should be near zero, too.
3. Without correction, when you shake gimbal, you see that the IMU_G_ERR changes significantly. With the correction applied, when you shake gimbal, IMU_G_ERR always stays near zero - it means that the external accelerations are compensated.
4. When you rotate frame relative to earth in all directions, or rotate camera relative to frame, the 3rd test is still passed correctly – it means that the ACC correction vector is translated to the main IMU sensor properly.

CMD_AHRS_HELPER (#56) – send or request attitude of the IMU sensor.

Use this command to provide a reference or replace the attitude estimated by the internal IMU sensor, by the attitude from a high-grade external IMU. The reasonable rate of sending this command is 20-50 Hz, maximum is 125 Hz. More information in the Appendix C: Providing external reference attitude/heading information from UAV

Name	Type	Min	Max	Possible values, remarks
MODE	1u or 2u**			**bit0**: 0 – get, 1 – set. **bit1**: location of the source IMU: 0 – camera platform, 1 – frame (modified by bits 8,9) **bit2**: if set, use as a reference only (any internal reference, if present, will be disabled). If not set, replaces the attitude and heading estimated by the internal sensor (both Z1 and H1 must be provided). **bit3**: if set, translate from camera to frame (or back) and use as a reference. **bit4,5**: selectively apply the correction by the provided Z and H vectors: `00` or `11` - use both Z and H vectors `01` - use Z vector only. `10` - use H vector only *for frw.ver. 2.69b5+:* **bit6,7**: selectively translate the correction (**bit3** should be set): `00` or `11` - translate both vectors `01` - translate Z vector only `10` - translate H vector only. *for 2-byte MODE (optional in frw.ver. 2.69b5+)**:* **bits8,9:** position of the reference system for the frame IMU correction: `00` – the same as frame IMU `01` – "on the frame" `10` – "below outer" (next to the outer motor in motor order counting from the frame) **bit10** *(frw.ver.2.70b1+):* disable external correction for the given IMU and vectors (specified in bits 1,4,5). Restore internal correction, if possible (using accelerometer for Z1, magnetometer for H1). REMARKS ON FLAGS *Bit1 encodes the position of the external source of attitude/heading information. When flag is set and frame IMU is*

				not enabled, it counts "on the frame". If frame IMU is enabled, it counts "in the frame IMU position", but can be modified by flags 8,9. *If Bit2 is set, attitude/heading is applied as a reference and the strength of correction depends on the parameters "Gyro trust" (inverted rule) and "Heading correction factor". The provided attitude and heading data replace the internal accelerometer and magnetometer sensor data, respectively.* *Bit3 is taken into account only if all motor angles are known from encoders or may be estimated using other ways. Bit 3 should be set if reference IMU is located on the frame and you need to correct main IMU located on the stabilized platform.* *Bits 4,5 may be used to selectively correct/replace only H (heading) or Z (attitude) vectors. For example, you can leave Z corrected by the internal accelerometer, and correct only H (heading).* *Bits 8,9 may be useful if system has the frame IMU enabled and the source of a reference attitude is located in a different position. For example, if the frame IMU is "below outer motor" but the external IMU is mounted on the gimbal's frame, before applying the correction we convert it to the coordinate system, linked to the frame IMU.* Below are some useful combinations of flags as an example: GET MODES `0x00` - request the main IMU attitude `0x02` - request the frame IMU attitude SET MODES `0x01` - use as a camera attitude (replaces the attitude estimated by the main IMU). `0x03` – replace the frame IMU attitude (if frame IMU is not enabled, use as gimbal's frame attitude). `0x05` - use as a reference for the main IMU. `0x07` - use as a reference for the frame IMU. `0x0B` – replace the frame IMU attitude/heading (or use as a gimbal's frame attitude if frame IMU is not enabled), translate to the camera coordinates and use as a reference for the main IMU. `0x0F` – use as a reference for the frame IMU, translate to the camera coordinates and use as a reference for the main IMU. `0x2F` – the same as above, but correct only a heading (use H-vector only). `0x012F` – external IMU is located on the frame, gimbal's frame IMU is located "below outer motor". After translation, use heading (H-vector) as a reference for the frame IMU and for the main IMU.
Z_VECT[3]	4f*3	-1.0f	1.0f	Unit vector that points down in END coordinate system (North-East-Down)* with the origin linked to the camera (MODE.bit1=0) or to the frame IMU or frame (MODE.bit1=1)
H_VECT[3]	4f*3	-1.0f	1.0f	Unit vector that points towards North in END coordinate system (North-East-Down)* with the same origin

* *Note that we use system END that differs from commonly used NED. See* *Appendix D: Coordinate system conversions*
** *MODE may be 1u or 2u for the extended flags supported starting from frw. ver. 2.69b5*

CMD_GYRO_CORRECTION (#75) – correct the gyroscope sensor's zero bias manually

Name	Type	Min	Max	Possible values, remarks
IMU_TYPE	1u			0 – main IMU, 1 – frame IMU
GYRO_ZERO_CORR[3]	2s*3			Zero offset for each axis in order X, Y, Z *Units: 0.001 gyro sensor unit*
GYRO_ZERO_HEADING_CORR	2s			Zero offset for global Z axis to correct a heading only. This correction is distributed to all axes automatically. *Units: 0.001 gyro sensor unit*

Responses

CMD_AHRS_HELPER (#56) – current attitude in vector form.

Name	Type	Min	Max	Possible values, remarks
Z_VECT[3]	4f*3	-1.0f	1.0f	Unit vector that points down in END coordinate system (North-East-Down)*
H_VECT[3]	4f*3	-1.0f	1.0f	Unit vector that points towards North in END coordinate system*

- *Note that we use system END that differs from commonly used NED. See* Appendix D: Coordinate system conversions
-

AHRS_DEBUG_INFO - information about the AHRS state

(frw.ver. 2.66+)

It's not a separate command. This structure is included as a part of other commands. Total size is 26 bytes.

Name	Type	Min	Max	Possible values, remarks
MAIN_IMU_REF_SRC	1u			Encodes the source of the reference information for the main IMU: bits 0..2: attitude reference source bits 3..5: heading reference source bit6: if set, internal sensor is connected and used; otherwise, AHRS information is set externally bit7: if set, the processing of this IMU is enabled Possible values for reference sources: `REF_NO = 0` - no reference `REF_INTERNAL = 1` - reference is provided by the internal sensor like accelerometer or magnetometer `REF_EXTERNAL = 2` - reference is set externally by the serial API or external IMU `REF_TRANSLATE = 3` - translate reference from other IMU (frame -> main, main -> frame)
FRAME_IMU_REF_SRC	1u			The same structure as for the main IMU.

MAIN_IMU_Z_REF_ERR	1u			Error between the reference defined by the MAIN_IMU_REF_SRC, and the estimated attitude *Units: 0.1°*
MAIN_IMU_H_REF_ERR	1u			Error between the reference defined by the MAIN_IMU_REF_SRC, and the estimated heading *Units: 0.1°*
FRAME_IMU_Z_REF_ERR	1u			Error between the reference defined by the FRAME_IMU_REF_SRC, and the estimated attitude *Units: 0.1°*
FRAME_IMU_H_REF_ERR	1u			Error between the reference defined by the FRAME_IMU_REF_SRC, and the estimated heading *Units: 0.1°*

EXT_IMU_STATUS	1u			bits 0..2 for status:
				`STATUS_DISABLED = 0`
				`STATUS_NOT_CONNECTED = 1`
				`STATUS_UNKNOWN = 2`
				`STATUS_ERROR = 3`
				`STATUS_BAD = 4`
				`STATUS_COARSE = 5`
				`STATUS_GOOD = 6`
				`STATUS_FINE = 7`
				(values 4..7 encode the quality of the attitude estimation)
				
				bits 3..7 for flags:
				`STATUS_FLAG_BAD_MAG = (1<<6) = 0x40`
				`STATUS_FLAG_NO_GPS_SIGNAL = (1<<7) = 0x80`

EXT_IMU_PACKETS_RECEIVED_CNT	2u	0	65535	
EXT_IMU_PARSE_ERR_CNT	2u	0	65535	
EXT_CORR_H_ERR	1u			Difference between the externally referenced heading and the current heading *Units: 0.1°*
EXT_CORR_Z_ERR	1u			Difference between the externally referenced attitude and the current attitude *Units: 0.1°*
RESERVED	13b			

CMD_EXT_IMU_DEBUG_INFO (#106) – debug information for the external IMU sensor

(frw.ver. 2.66+)

Name	Type	Min	Max	Possible values, remarks
AHRS_DEBUG_INFO	26b			See the AHRS_DEBUG_INFO specification
DCM	9*4f	-1.0f	1.0f	Rotation matrix (DCM) received from the external IMU and converted to the (END) (East-North-Down) coordinates.
ACC_BODY	3*4f			Linear acceleration (with the gravity vector subtracted) in sensor's local coordinates.

Controlling gimbal movements

Requests

CMD_CONTROL (#67) – controls gimbal movement

Name	Type	Min	Max	Possible values, remarks
Legacy format: mode is common for all axes				

CONTROL_MODE	1u			Bits 0..3 for mode, bits 4..7 for flags.
				
				Modes:
				
				`MODE_NO_CONTROL=0`
				Finish serial control and restore normal RC control.
				
				`MODE_IGNORE=7` *(frw.ver.2.70b4)*
				Ignore this axis and all parameters, keeping it in the actual state
				
				`MODE_SPEED=1`
				Gimbal travels with the given speed in the Euler coordinates until the next `CMD_CONTROL` command comes. Given angle is ignored.
				
				`MODE_ANGLE=2`
				`MODE_ANGLE_SHORTEST=8` *(frw.ver.2.70b7)*
				Gimbal travels to the given Euler angle with the automatically calculated speed according to the acceleration limit. The nominal speed may be provided by the **SPEED** parameter. Speed is additionally attenuated near target to keep the control smooth (if `CONTROL_FLAG_TARGET_PRECISE` is not set).
				• `MODE_ANGLE:` gimbal travels to the new absolute setpoint angle, making multiple turns if needed, but limiting a rotation by ±720° per command due to limited 16bit parameter range.
				• `MODE_ANGLE_SHORTEST:` gimbal travels to the point on a 360° circle by the shortest path, taking into account the software limits of encoders when choosing a direction (so it can go by the long path if the short path is blocked)
				
				`MODE_SPEED_ANGLE=3`
				Gimbal travels with the given **SPEED** parameter. Additionally, controller keeps the given angle and fixes the accumulated error by adjusting the actual speed in a direction of error minimization, defined by the “Outer P” GUI parameter. This error may appear because the estimated target angle (integral of `SPEED` by dt) may differ from the actual target angle, because the actual target speed is internally filtered by LPF and acceleration limit, if they are enabled. This mode should be chosen when it's required to exactly repeat the rotation of the user-operated device (like joystic or wheel), precisely tracking its speed and angle.
				
				`MODE_RC=4`
				The **ANGLE** parameter is used as RC signal and overrides any other signal source, assigned to this

				axis. Normal working range is -500..500. A special value -10000 encodes a "signal lost" condition.
				The flag `CONTROL_FLAG_AUTO_TASK` can affect this mode (see below).
				*Prior to 2.61 frw. ver., '**SPEED**' parameter is ignored.*
				
				`MODE_RC_HIGH_RES=6` *(frw. ver. 2.66b2)*
				The same as the MODE_RC, but the range of the ANGLE parameter has better resolution: -16384..16384. A special value -32768 encodes a "signal lost" condition.
				
				`MODE_ANGLE_REL_FRAME=5`
				First, the neutral point of a camera relative to a frame is found in the Euler coordinates for a given axis. Than, the given angle (in ±360° range) is added to this point, and camera travels to it. Note that the given angle does not relate to a particular motor, it relates to global Euler angles!
				
				Flags:
				
				`CONTROL_FLAG_AUTO_TASK=(1<<6)`
				Firmware version: 2.62b7
				*Applicable for: **MODE_ANGLE, MODE_ANGLE_SHORTEST, MODE_ANGLE_REL_FRAME***
				The task is processed with the speed and acceleration configured for *automated tasks.* If the **SPEED** parameter is provided, it's used instead. When all target angles are reached with the 1-degree tolerance, confirmation is sent: `CMD_CONFIRM(CMD_CONTROL, 1).`
				Use this flag to move gimbal to a certain position as fast as possible, and receive confirmation when the target is reached. If system can't process the planned trajectory for some reasons, motion will be interrupted after 10-second timeout.
				
				`CONTROL_FLAG_FORCE_RC_SPEED=(1<<6)`
				Firmware version: 2.62b7
				*Applicable for: **MODE_RC***
				This flag forces a control in the "SPEED" mode, with the dead-band, trimming and inversion settings are NOT applied to the provided RC signal, but the LPF, Expo curve and ACC limiter are still applied. Use this flag to control gimbal from remote applications, where signal is well-defined and you need to have a direction of rotation that does not depend on gimbal's "Inverse" and "Mode" parameters.
				
				`CONTROL_FLAG_HIGH_RES_SPEED=(1<<7)`
				Firmware version: 2.60b0
				Applicable for: all modes
				Speed units changed to 0.001 deg/sec for extremely slow motion (like timelapse shooting)
				
				`CONTROL_FLAG_TARGET_PRECISE=(1<<5)`
				Firmware version: 2.70b1
				*Applicable for: **MODE_ANGLE, MODE_ANGLE_SHORTEST, MODE_ANGLE_REL_FRAME***
				If this flag is set, the speed is not decreased in a vicinity of target. It allows to get more predictive speed profile for the motion trajectory. If not set, actual speed is decreased near target to smooth over the jerks when

				distance to target is small and target is updated frequently by small steps. `CONTROL_FLAG_MIX_FOLLOW=(1<<4)` *Firmware version: 2.70b5* *Applicable for: `MODE_SPEED, MODE_ANGLE, MODE_ANGLE_SHORTEST`* If this flag is set, the follow mode is not overridden, but is mixed with the commanded motion, like it happens for the regular RC control in SPEED or ANGLE mode. If this flag is not set, the commanded motion completely overrides the follow control for this axis.
Extended format (firmware ver. 2.55b5+): mode is set independently for each axes				
CONTROL_MODE[3]	1u*3			see definition above
The remaining part is common for all formats				
axis = (1..3) SPEED	2s	- - -	- - -	Speed of rotation. Overrides the speed settings in the GUI Notes: • If the acceleration limit is enabled in the settings, the actual speed is filtered by it for all modes. • Additionally, the actual speed is decreased near target to prevent step-wise motion, unless flag `CONTROL_FLAG_TARGET_PRECISE` is set. *`MODE_ANGLE, MODE_RC, MODE_ANGLE_REL_FRAME:`* • the value is always positive and may be set to 0 - in this case, speed is taken from the RC settings (or from the “Automated tasks” settings if `CONTROL_FLAG_AUTO_TASK` is set): *`SPEED = settings.RC_SPEED*16`* *`MODE_SPEED, MODE_SPEED_ANGLE:`* • the values is signed (negative for opposite direction) *Units: 0,1220740379 deg./sec.* *0.001 deg./sec., if `CONTROL_FLAG_HIGH_RES_SPEED` is set*
ANGLE	2s	-32768	32767	Depends on the `MODE` parameter: • `MODE_ANGLE, MODE_SPEED_ANGLE`: encodes the absolute target angle. If `CONTROL_FLAG_MIX_FOLLOW` is set, angle is considered relative to the point where mode starts. • `MODE_SPEED`: ignored • `MODE_RC`: encodes RC signal in range -500..500 • `MODE_RC_HIGH_RES`: encodes RC signal in range -16384..16384 *Units: 0,02197265625 degree.*

Notes:

- When `CMD_CONTROL` comes first time, it overrides the regular RC or Follow mode control, unless `CONTROL_FLAG_MIX_FOLLOW` is set. To switch back to a normal mode, send this command with the `MODE=0` for all axes, and all data set to zeros. All parameters that were changed by the `CMD_CONTROL_CONFIG`, will be restored to their default values.
  Behavior is different if `CONTROL_FLAG_AUTO_TASK` is set: when the commanded motion is finished, system returns to a normal mode automatically.
- The optimal rate of sending this command is 50..125 Hz. If the rate of `CMD_CONTROL` command is lower, use a low-pass filtering to prevent step-wise response. It can be set by the command `CMD_CONTROL_CONFIG`

separately for SPEED and ANGLE parameters, with the rule: the lower the rate, the more filtering is required.

- Confirmation is sent on each CMD_CONTROL command unless CONTROL_CONFIG_FLAG_NO_CONFIRM is set. Additional confirmation is sent when the target angle is reached, if CONTROL_FLAG_AUTO_TASK is set.
- Automated tasks has greater priority then CMD_CONTROL. For example, executing menu command "Level ROLL to horizon" overrides CMD_CONTROL for ROLL axis until task is finished.
- The ANGLE rage is limited by ±720°. It's possible to get rid of this limitation by sending multiple commands in a sequence: when gimbal approaches the current setpoint, send a new setpoint with the ANGLE computed using natural C++ integer arithmetics with wrap-around overflow. Example: to make 3 full turns (1080°) the first command sends axis to 720° (ANGLE=32767) and the second command sends it to 1080° (ANGLE=-16384, which is computed as 32767+16384 in int16_t variable).
  Starting from firmware 2.70b8, it's possible to send several commands in sequence immediately one after another (the only condition is that the distance between two setpoints does not exceed 720° or 32768), or use CMD_CONTROL_EXT were 20bit ANGLE parameter does not have such limitation and allows to command a rotation up to 4096 full turns in a single command.
- The ANGLE parameter doesn't always match the TARGET_ANGLE variable (reported in CMD_ANGLES, CMD_REALTIME_DATA and others), which means “the setpoint angle that gimbal should keep right now”. Only after the motion sequence is finished, they match (excepting cases when CONTROL_MODE and flags change the resulting setpoint before the motion starts, like “travel shortest distance” or “mix with the follow mode”).
- This command is developing permanently: new modes and flags are added and behavior is improved, so old firmware versions may handle this mode differently then described in this document.
- See the Appendix A for a source code examples

CMD_CONTROL_EXT (#121) – controls gimbal movement, extended version

(frw. ver. 2.68+)

An extended version of CMD_CONTROL. It allows to omit unused fields to save the bandwidth. The content of the command depends on enabled bits in DATA_SET parameter.

Name		Type	Min	Max	Possible values, remarks
DATA_SET		2u			Defines which data is provided in this command with a variable length. Bits are grouped by axes [0..4], [5..9], [10..14] In each group: bit0: SPEED parameter is given bit1: ANGLE parameter is given bit2: 4 bytes ANGLE with increased resolution bit3: 4 bytes SPEED with increased resolution bit4: reserved
axis =(1..3)	CONTROL_MODE	1u			See the CONTROL_MODE parameter in CMD_CONTROL. If neither SPEED or ANGLE bits are enabled, the CONTROL_MODE parameter should be omitted for this axis
	MODE_FLAGS	1u			reserved
	SPEED	2s 4s*			See the SPEED parameter in CMD_CONTROL ** (frw.ver. 2.72b0+) If bit3 in DATA_SET is set, SPEED is 32bit signed value with the increased resolution 0.001 deg./sec. Otherwise, it is 16bit signed value with resolution 0,1220740379 deg./sec.*

	ANGLE	2s 4s*			See the ANGLE parameter in CMD_CONTROL **If bit2 in DATA_SET is set, ANGLE is encoded as 32-bit signed value with the resolution 20 bits per turn. Otherwise, it's 16-bit signed value with the resolution 14 bits per turn.* *Units: 0,00034332275390625 or 0,02197265625 degrees*

Confirmation (CMD_CONFIRM) is sent in response, unless it's disabled in CMD_CONTROL_CONFIG. Error (CMD_ERROR) is sent in response if format is incorrect (for example, payload size differs from expected).

Examples (payload only):

`00 1C 02 00 F4 01 00 00 30 00` – rotate YAW to 1080° (MODE_ANGLE, 20bit), SPEED=61°/sec
`00 0C 02 00 F4 01 00 10` – rotate YAW to 90° (MODE_ANGLE,14bit), SPEED=61°/sec
`42 08 02 00 00 00 02 00 00 00 02 00 00 00` – move all axes to 0° with the default speed

CMD_CONTROL_CONFIG (#90) – configure the handling of CMD_CONTROL command

(frw. ver. 2.61+)

Name		Type	Min	Max	Possible values, remarks
TIMEOUT_MS		2u	0	65535	0 - disable timeout >0 - if no CMD_CONTROL command will come in a given time on any channel, serial control will be finished. Default value after startup is 0 (no timeout). *Units: ms*
CH1_PRIORITY CH2_PRIORITY CH3_PRIORITY CH4_PRIORITY THIS_CH_PRIORITY		1u*5	0	255	Channels are counted in order: UART1, RC_SERIAL, UART2, USB_VCP (how they are named in the User Manual). THIS_CH means current port, where command is sent. Values: 0 - do not change the priority 1..255 - set the priority of a given channel. In case of concurrent CMD_CONTROL commands, they will be accepted only on a channel that has higher or equal priority than others. *Default value is 0 for all channels after startup.*
axis = (1..3)	ANGLE_LPF	1u	0	15	LPF factor for filtering the 'ANGLE' parameter in the modes "MODE_ANGLE", "MODE_SPEED_ANGLE". Helps to mitigate a step-wise response if update rate is low. *Default value is 0 – no filtering is applied.*
	SPEED_LPF	1u	0	15	LPF factor for filtering the 'SPEED' parameter in the modes "MODE_SPEED", "MODE_SPEED_ANGLE". Helps to mitigate a step-wise response if update rate is low. *Default value is 0 – no filtering is applied.*
	RC_LPF [1)]	1u	0	255	LPF factor for filtering RC signal in the mode "MODE_RC", “MODE_RC_HIGH_RES”. Helps to mitigate a step-wise response if update rate is low. 0 – do not change.
	ACC_LIMIT [1)] *(frw.ver. 2.70b1)*	2u	0	65535	Acceleration limiter filter, applied to speed profile in all control modes. 0 – do not change >0 – it overrides the default value from system settings.

					Units: degrees/sec²
	JERK_SLOPE [1)] *(frw.ver. 2.70b8+)*	1u	0	255	The rate of change of an acceleration, forming S-curve in a speed profile, increasing the smoothness of control. This parameter defines the time required to rise the acceleration from zero to a nominal value. *Special values:* *0 – ignore this parameter* *1 – disable jerk slope function* *Units: 20ms*
	RESERVED	1b			
RC_EXPO_RATE [1)]		1u	0	100	Exponential curve for filtering RC signal in the mode "MODE_RC". *0 – do not change*

FLAGS [1)]		2u			`FLAG_NO_CONFIRM=(1<<0)`
					(frw.ver. 2.66b2)
					If set, controller does not send confirmation on each CMD_CONTROL command.
					
					`FLAG_SERVO_MODE_ENABLE=(1<<1)`
					`FLAG_SERVO_MODE_DISABLE=(1<<2)`
					(frw.ver. 2.72b0)
					Enable or disable servo mode – in which the commanded rotation is done around motors' axes rather than Euler axes; ANGLE is relative to the calibrated neutral point for each motor.

EULER_ORDER [1)] *(frw.ver. 2.72b0)*		1u			0: don't change 1: Cam - PITCH - ROLL - YAW 2: Cam - ROLL - PITCH - YAW 3: Cam - PITCH(M) - ROLL – YAW(M) 4: Cam - ROLL - PITCH(M) - YAW(M) 5: Cam - YAW - ROLL – PITCH 10: Screen-related X,Y,Z
RESERVED		9b			

Confirmation is sent on success.

[1)] *All changes in these parameters valid only while serial control is active. When serial control finishes, the GUI-configured values are restored.*

CMD_API_VIRT_CH_CONTROL (#45) – update a state of 32 virtual channels.

Named as “API_VIRT_CHxx” in the GUI; may be assigned as RC source to any task.

Name	Type	Min	Max	Possible values, remarks
API_VIRT_CH1 … API_VIRT_CH32	2s*32	-500	500	Value may go slightly outside these limits. Use a special value “-10000” to mark that channel has “undefined” state (its treated as “signal lost” like with the regular RC inputs)

** Starting from firmware 2.70b5, any number of channels (1..32) may be provided, to save bandwidth by omitting unused channels.*

CMD_API_VIRT_CH_HIGH_RES (#116) – update a state of 32 virtual channels

(frw.ver. 2.68b7+)

Named as “API_VIRT_CHxx” in the GUI; may be assigned as RC source to any task.
This command is similar to CMD_API_VIRT_CH_CONTROL, excepting it has higher resolution.

Name	Type	Min	Max	Possible values, remarks
API_VIRT_CH1 … API_VIRT_CH32*	2s	-16384	16384	Value may go slightly outside these limits. Special value -32768 sets channel to undefined state (input is ignored)

** Any number of channels (1..32) may be provided, to save bandwidth by omitting unused channels.*

CMD_CONTROL_QUAT (#140) – control gimbal position in quaternions

(frw.ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks

MODE	1u			`MODE_DISABLED = 0`
				Stops this type of control and return to a regular operation mode
				
				`MODE_SPEED = 1`
				Gimbal rotates with the TARGET_SPEED. The TARGET_ATTITUDE is ignored and should be omitted. No motion profile is applied in this mode, excepting the low-pass filtering, if enabled in `CMD_CONTROL_QUAT_CONFIG`
				
				`MODE_ATTITUDE = 2`
				Gimbal moves to the TARGET_ATTITUDE with the configured speed profile. TARGET_SPEED is ignored and should be omitted.
				
				`MODE_SPEED_ATTITUDE = 5`
				Gimbal rotates with the TARGET_SPEED and if the actual position differs from the provided TARGET_ATTITUDE, it’s corrected in order to match.
				
				`MODE_SPEED_LIMITED = 9`
				The same as MODE_SPEED but the motion profile (acceleration limit) is applied

FLAGS	1u			`FLAG_CONFIRM = (1<<0)`
				If set, CMD_CONFIRM is sent in response.
				
				`FLAG_ATTITUDE_PACKED = (1<<1)`
				If set, TARGET_ATTITUDE is packed to 8 bytes (see Appendix F for format description)
				
				`FLAG_ATTITUDE_LIMITED_180 = (1<<2)`
				If set, TARGET_ATTITUDE is considered to be limited by ±180° and a ‘phase unwrap’ filter is applied by comparing the new value to the previous value and choosing the shortest path. See Quaternions section in Appendix D for more details
				
				`FLAG_AUTO_TASK=(1<<6)`
				Applicable for: `MODE_ATTITUDE`
				`NOT IMPLEMENTED`

				The task is processed with the speed and acceleration configured for *automated tasks*. When the target attitude is reached with the 1-degree tolerance, confirmation is sent: `CMD_CONFIRM(CMD_CONTROL_QUAT, 1)`, then gimbal switches to a normal operation mode. Use this flag to move gimbal to a certain position as fast as possible, and receive confirmation when the target is reached. If system can't process the planned trajectory for some reasons, motion will be interrupted after 10-second timeout.
TARGET_ATTITUDE[4]*	4f*4			Target attitude in a form of unit quaternion [w, x, y, z]
TARGET_SPEED[3]**	4f*3			Target speed in a form of rotation vector [x, y, z] *Units: rad/sec.*

** Provided only in **`MODE_ATTITUDE, MODE_SPEED_ATTITUDE`***
*** Provided only in **`MODE_SPEED, MODE_SPEED_ATTITUDE, MODE_SPEED_LIMITED`***

NOTE: Rotation vector and quaternion are in END coordinate system. See Coordinate system conversions.

No confirmation is sent unless FLAG_CONFIRM is set. In case of wrong format or wrong gimbal state, CMD_ERROR is sent in response.

In order to query the real-time state related to this type of control, use CMD_CONTROL_QUAT_STATUS (#141)

CMD_CONTROL_QUAT_CONFIG (#142) – configure the quaternion-based control mode

(frw. ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks

DATA_SET	2u			Bit mask defining which parameters are updated – only these fields should be included into the payload.
				
				`bit0: MAX_SPEED[3]`
				`bit1: ACC_LIMIT[3]`
				`bit2: JERK_SLOPE[3]`
				`bit3: FLAGS`
				`bit4: ATTITUDE_LPF_FREQ`
				`bti5: SPEED_LPF_FREQ`

MAX_SPEED[3]	2u*3			Motion profile max. speed limit for [X,Y,Z] axes. *Units: 0,1220740379 deg./sec.* *Default value is taken form RC settings*
ACC_LIMIT[3]	2u*3			Motion profile acceleration limit for [X,Y,Z] axes. 0 – disable *Units: degrees/sec^2* *Default value is taken form RC settings*
JERK_SLOPE[3]	2u*3			The rate of change of an acceleration for [X,Y,Z] axes. This parameter defines the time required to rise the acceleration from zero to a nominal value. Increases the smoothness of control. 0 – disable *Units: ms* *Default value is taken form RC settings*

FLAGS	2u			`FLAG_MOTION_PROFILE_SPLIT_XYZ = (1<<0)` *Applicable for: **`MODE_ATTITUDE`**, **`MODE_SPEED_LIMITED`**,* *Default value: **disabled*** If set, the motion profile is applied separately for X,Y,Z axes according to the acceleration limit settings for ROLL (Y), PITCH (X), YAW (Z) axes. If not set, the motion profile is applied to the speed along the shortest path trajectory. Configuration is taken from the X axis (PITCH in Euler settings)
ATTITUDE_LPF_FREQ	1u			LPF cut-off frequency for TARGET_ATTITUDE and TARGET_SPEED parameters correspondingly. Helps to mitigate a step-wise response if update rate is low. 0 – disable LPF *Units: 1Hz* *Default value: disabled*
SPEED_LPF_FREQ	1u			

Confirmation is sent on success.

Controlling extra auxiliary motors

Extra motors based on CAN_DRV are supported in the “Extended” family of controllers.

Requests

CMD_EXT_MOTORS_ACTION (#128) – execute an action on the selected motor(s)

(frw. ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks
FOR_MOTORS	1u			Bits 0...6 select the motors this command affects Bit7 – if set, confirmation is sent in response

ACTION	1u			`MOTOR_OFF_FLOATING = 1`
				turn off and leave floating
				
				`MOTOR_OFF_BRAKE = 2`
				turn off and brake (default)
				
				`MOTOR_OFF_SAFE = 3`
				compensate for external force to slow down the rotation, then turn off with brake
				
				`MOTOR_ON = 4`
				turn on
				
				`HOME_POSITION = 5`
				move to home position
				
				`SEARCH_HOME = 6`
				search home period for multiturn or geared motors

ARGS	?			Reserved. Optional arguments, depending on action.

CMD_EXT_MOTORS_CONTROL (#129) – control the selected motor(s)

(frw. ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks
FOR_MOTORS	1u			Bits 0...6 select the motors this command affects. The setpoint is provided for each motor individually according to the DATA_SET format. Bit7 – if set, confirmation is sent in response
DATA_SET	1u			Defines which data is provided in this command: `bit0:` if set, SETPOINT is 32bit, otherwise 16bit `bit1:` PARAM1 is provided in 16bit `bit2:` PARAM1 is provided in 32bit
The meaning of SETPOINT and PARAM (optional) depends on the configured mode for each motor. *The list size is defined by the number of selected bits in FOR_MOTORS variable.*				

Motor = (1..7)	SETPOINT16	2s			Torque mode: ±32767 for the maximum torque the driver can provide Speed mode: *units 0,1220740379 deg./sec.* Position mode: *14 bits for a turn, units 0,02197265625 deg. There is an unwrap filter for continued rotation properly handling integer overflow.*
	SETPOINT32	4s			Torque mode: not applicable Speed mode: *units micro-radians/sec.* Position mode: *resolution 20 bits per turn, units 0,00034332275390625 deg.*
	PARAM1	2s or 4s			Speed mode: overrides the internally calculated angle (as if both SPEED and ANGLE setpoints were provided externally) Position mode: overrides the configured max. speed

In order to query the extra motor real-time state, use CMD_EXT_MOTORS_STATE (#131)

CMD_EXT_MOTORS_CONTROL_CONFIG (#130) – configure run-time parameters for the selected motor(s)

(frw. ver. 2.73.x)

Name	Type	Min	Max	Possible values, remarks
FOR_MOTORS	1u			Bits 0...6 select the motors this command affects. The configured parameters are common for all selected motors. Bit7 – if set, confirmation is sent in response

DATA_SET	2u			Bit mask defines the variables to be updated. If any bit is not set, the corresponding variables should be omitted.
				`bit0: MODE`
				`bit1: MAX_SPEED`
				`bit2: MAX_ACCELERATION`
				`bit3: JERK_SLOPE`
				`bit4: MAX_TORQUE`

MODE	1u			0: position 1: speed 2: torque
MAX_SPEED	2u			Applied for all modes *Units: 2 deg./sec.*
MAX_ACCELERATION	2u			Applied for speed and position modes *Units: 2 deg./sec^2*
JERK_SLOPE	2u			Applied for speed and position modes *Units: ms*
MAX_TORQUE	2u	0	65535	Applied for all modes *Units: Relative to the max. available torque*

Miscellaneous commands

Requests

CMD_RESET (#114) – reset device

Simple format: no parameters. Resets the device without delay and confirmation

Extended format:

Name	Type	Min	Max	Possible values, remarks
FLAGS	1u			bit0 – if set, CMD_RESET will be sent to the host as a confirmation. bit1 – if set, back up some state variables and restore them after restart: – motors ON/OFF state – setpoint angles – follow mode offset angles
DELAY_MS	2u			After confirmation is sent, waits for a given time (in ms) before reset.

CMD_BOOT_MODE_3 (#51) – enter bootloader mode to upload firmware

Simple format: no parameters. Enters boot mode without delay and confirmation

Extended format:

Name	Type	Min	Max	Possible values, remarks
CONFIRM	1u			0 – no confirmation 1 - command CMD_RESET will be sent back for confirmation
DELAY_MS	2u			After confirmation is sent, waits for a given time (in ms) before reset. External application can free up resources and properly close the serial connection before controller enters boot mode.

CMD_TRIGGER_PIN (#84) - trigger output pin

Name	Type	Min	Max	Possible values, remarks
PIN_ID	1u			Triggers pin only if it is not used for input RC_INPUT_ROLL = 1 RC_INPUT_PITCH = 2 EXT_FC_INPUT_ROLL = 3 EXT_FC_INPUT_PITCH = 4 RC_INPUT_YAW = 5 PIN_AUX1 = 16 PIN_AUX2 = 17 PIN_AUX3 = 18 PIN_BUZZER = 32 PIN_SSAT_POWER** = 33

				** PIN_SSAT_POWER triggers 3.3V power line in the Spektrum connector (low state enables power)
STATE	1u			`LOW = 0 (GND)` - pin can sink up to 40mA `HIGH = 1 (+3.3V)` - pin can source up to 40mA `FLOATING = 2` *(frw. ver. 2.66+)*

Confirmation is sent only if pin is not occupied for other functions and was really triggered.

CMD_MOTORS_ON (#77) - switch motors ON

No parameters. Confirmation is sent in response.

CMD_MOTORS_OFF (#109) - switch motors OFF

Name	Type	Min	Max	Possible values, remarks
MODE *(frw.ver. 2.68b7+)*	1u			0 – normal mode: turn motors OFF leaving driver in a high impedance; 1 – "break mode": turns motors OFF leaving driver in a low impedance; 2 – "safe stop" mode for unbalanced gimbals: reduce power and wait while all motors stop rotating, then power OFF completely.

Confirmation is sent in response.

CMD_EXECUTE_MENU (#69) - execute menu command

Name	Type	Min	Max	Possible values, remarks

CMD_ID	1u			`MENU_CMD_NO = 0`
				`MENU_CMD_PROFILE1 = 1`
				`MENU_CMD_PROFILE2 = 2`
				`MENU_CMD_PROFILE3 = 3`
				`MENU_CMD_SWAP_PITCH_ROLL = 4`
				`MENU_CMD_SWAP_YAW_ROLL = 5`
				`MENU_CMD_CALIB_ACC = 6`
				`MENU_CMD_RESET = 7`
				`MENU_CMD_SET_ANGLE = 8`
				`MENU_CMD_CALIB_GYRO = 9`
				`MENU_CMD_MOTOR_TOGGLE = 10`
				`MENU_CMD_MOTOR_ON = 11`
				`MENU_CMD_MOTOR_OFF = 12`
				`MENU_CMD_FRAME_UPSIDE_DOWN = 13`
				`MENU_CMD_PROFILE4 = 14`
				`MENU_CMD_PROFILE5 = 15`
				`MENU_CMD_AUTO_PID = 16`
				`MENU_CMD_LOOK_DOWN = 17`
				`MENU_CMD_HOME_POSITION = 18`
				`MENU_CMD_RC_BIND = 19`
				`MENU_CMD_CALIB_GYRO_TEMP = 20`
				`MENU_CMD_CALIB_ACC_TEMP = 21`
				`MENU_CMD_BUTTON_PRESS = 22`
				`MENU_CMD_RUN_SCRIPT1 = 23`
				`MENU_CMD_RUN_SCRIPT2 = 24`

				MENU_CMD_RUN_SCRIPT3 = 25 MENU_CMD_RUN_SCRIPT4 = 26 MENU_CMD_RUN_SCRIPT5 = 27 MENU_CMD_CALIB_MAG = 33 MENU_CMD_LEVEL_ROLL_PITCH = 34 MENU_CMD_CENTER_YAW = 35 MENU_CMD_UNTWIST_CABLES = 36 MENU_CMD_SET_ANGLE_NO_SAVE = 37 MENU_HOME_POSITION_SHORTEST = 38 MENU_CENTER_YAW_SHORTEST = 39 MENU_ROTATE_YAW_180 = 40 MENU_ROTATE_YAW_180_FRAME_REL = 41 MENU_SWITCH_YAW_180_FRAME_REL = 42 MENU_SWITCH_POS_ROLL_90 = 43 MENU_START_TIMELAPSE = 44 MENU_CALIB_MOMENTUM = 45 MENU_LEVEL_ROLL = 46 MENU_REPEAT_TIMELAPSE = 47 MENU_LOAD_PROFILE_SET1 = 48 MENU_LOAD_PROFILE_SET2 = 49 MENU_LOAD_PROFILE_SET3 = 50 MENU_LOAD_PROFILE_SET4 = 51 MENU_LOAD_PROFILE_SET5 = 52 MENU_LOAD_PROFILE_SET_BACKUP = 53 MENU_INVERT_RC_ROLL = 54 MENU_INVERT_RC_PITCH = 55 MENU_INVERT_RC_YAW = 56 MENU_SNAP_TO_FIXED_POSITION = 57 MENU_CAMERA_REC_PHOTO_EVENT = 58 MENU_CAMERA_PHOTO_EVENT = 59 MENU_MOTORS_SAFE_STOP = 60 MENU_CALIB_ACC_AUTO = 61 MENU_RESET_IMU = 62 MENU_FORCED_FOLLOW_TOGGLE = 63 MENU_AUTO_PID_GAIN_ONLY = 64 MENU_LEVEL_PITCH = 65 MENU_MOTORS_SAFE_TOGGLE 66 MENU_TIMELAPSE_STEP1 67 MENU_EXT_GYRO_ONLINE_CALIB 68 MENU_DISABLE_FOLLOW_TOGGLE 69 MENU_SET_CUR_POS_AS_HOME 70 MENU_STOP_SCRIPT 71 MENU_TRIPOD_MODE_OFF 72 MENU_TRIPOD_MODE_ON 73 MENU_SET_RC_TRIM 74 MENU_HOME_POSITION_MOTORS 75 MENU_RETRACTED_POSITION 76 MENU_SHAKE_GENERATOR_OFF 77 MENU_SHAKE_GENERATOR_ON 78

CMD_AUTO_PID (#35) – Starts automatic PID calibration

(frw. ver. prior to 3.00)

Name	Type	Min	Max	Possible values, remarks
PROFILE_ID	1u			switch to this profile before start of the calibration and save result there
CFG_FLAGS	1u			AUTO_PID_CFG_ROLL = 1 AUTO_PID_CFG_PITCH = 2 AUTO_PID_CFG_YAW = 4

				`AUTO_PID_CFG_SEND_GUI = 8`
				- if set, sends a progress of tuning to the GUI in the CMD_AUTO_PID after each iteration
				`AUTO_PID_CFG_KEEP_CURRENT = 16`
				- if set, starts from existing settings. If not set, starts from zero
				`AUTO_PID_CFG_TUNE_LPF_FREQ = 32`
				- if set, tunes LPF filters, too
				`AUTO_PID_CFG_ALL_PROFILES = 64`
				- if set, updates tuned parameters in all profiles. Otherwise, updates only the selected profile.

GAIN_VS_STABILITY	1u	0	255	0 - better stability, 255 - better tracking of a reference
MOMENTUM	1u	0	255	0 - detect automatically, 1 - low weight and strong motor, 255 - big weight and weak motor
ACTION	1u			0 – start tuning
RESERVED	14b			

- On start, a confirmation is sent in the command CMD_CONFIRM(CMD_AUTO_PID).
- When finished, the controller sends a full set of tuned parameters to the GUI (CMD_READ_PARAMS_XX), for the selected or for all profiles.
- To interrupt currently running auto-tuning process, send this command with zero values in all fields.

CMD_AUTO_PID2 (#108) – Starts automatic PID calibration ver.2

(frw. ver. 3.00+)

Name		Type	Min	Max	Possible values, remarks
ACTION		1u			ACTION_START=1 start tuning (do not update config in EEPROM) ACTION_START_SAVE=2 save config to EEPROM and start tuning ACTION_SAVE=3 save config to EEPROM ACTION_STOP=5 stop tuning ACTION_READ=6 read config from EEPROM
RESERVED		10b			
The following data is required only for ACTION_START, ACTION_START_SAVE:					
CFG_VERSION		1u			version 1
for axis = (1..3)	AXIS_FLAGS	1u			bit0: this axis is enabled bit1: tune LPF bits2..3: number of notch filters to tune, 0-3
	GAIN	1u	0	255	stability vs performance ratio
	STIMULUS	1u	0	255	stimulus signal strength
	EFFECTIVE_FREQ	1u	0	255	Effective frequency, Hz
	PROBLEM_FREQ	1u	0	255	Problematic frequency, Hz
	PROBLEM_MARGIN	1u	0	255	Problematic margin, dB*10

RESERVED	6b			
GENERAL_FLAGS	2u			bit0: start from current values bit1: save result to all profiles bit2: tune gain only bit3: reserved bit4: auto-save
STARTUP_CFG	1u			Automatically run at system startup 0 - Disabled 1 - Tune gain only 2 - Tune all parameters
RESERVED	22b			

Confirmation is sent immediately in the command CMD_CONFIRM(CMD_AUTO_PID2).
If error is detected in parameters, CMD_ERROR is sent with the error code:

1: read from EEPROM failed (data is corrupted or empty)
2: can't run algorithm at this moment
3: write to EEPROM failed
4: unknown action
5: wrong command size

When finished, the controller sends a full set of tuned parameters to the GUI (CMD_READ_PARAMS_XX), for the current profile.

CMD_SERVO_OUT (#36) – Output PWM signal on the servo1..4 pins

Name	Type	Min	Max	Possible values, remarks
SERVO_TIME[4]*	2s*4	-1	2500	value < 0: free up this pin and make it floating value = 0: configure this pin as output and set it to 'Low' state value > 0: PWM pulse time, us. Should be less than PWM period, configured by the “SERVO_RATE” parameter. Regular servo accept values in range about 500..2500 us, 1500 us is neutral position, PWM period is 20000 us or less. *frw.ver 2.70b8+:* in a special PWM duty cycle output mode, value 1000 corresponds to 0% duty cycle, value 2000 to 100% duty cycle.
RESERVED	8b			

Servo mode is available on the ports:

Servo1 - EXT_FC_ROLL
Servo2 - EXT_FC_PITCH
Servo3 - RC_PITCH 3
Servo4 - AUX1

CMD_I2C_WRITE_REG_BUF (#39) – writes data to any device connected to I2C line

Name	Type	Min	Max	Possible values, remarks
DEVICE_ADDR	1u			bit0: I2C port 0 for external port (IMU sensor is connected) 1 for internal port (EEPROM) bit1..7: I2C address

REG_ADDR	1u			register to write
DATA	?			remaining bytes are counted as data

On successful writing, confirmation CMD_CONFIRM is sent in response.

CMD_I2C_READ_REG_BUF (#40) – requests reading from any device connected to I2C line

Name	Type	Min	Max	Possible values, remarks
DEVICE_ADDR	1u			bit0: I2C port 0 for external I2C port 1 for internal I2C port (where on-board I2C devices are connected) bit1..7: I2C address*
REG_ADDR	1u			register to read (only 1-byte addressing is supported)
DATA_LEN	1u			length of data to read

On successful reading, CMD_I2C_READ_REG_BUF command is sent in response.

** for example, to read from ICM-20602 on address 0x68 from 'who am I' register, payload is "D0 75 01"*

CMD_RUN_SCRIPT (#57) – start or stop user-written script

Name	Type	Min	Max	Possible values, remarks
MODE	1u			0 – stop 1 – start 2 – start with debug information is sent back in the CMD_SCRIPT_DEBUG
SLOT	1u	0	4	slot number, starting from 0.
RESERVED	32b			

CMD_BEEP_SOUND (#89) – play melody by motors or emit standard beep sound

Name	Type	Min	Max	Possible values, remarks

MODE	2u			Pre-defined melodies:
				`BEEPER_MODE_CALIBRATE = (1<<0)`
				`BEEPER_MODE_CONFIRM = (1<<1)`
				`BEEPER_MODE_ERROR = (1<<2)`
				`BEEPER_MODE_CLICK = (1<<4)`
				`BEEPER_MODE_COMPLETE = (1<<5)`
				`BEEPER_MODE_INTRO = (1<<6)`
				Custom melody:
				`BEEPER_MODE_CUSTOM_MELODY = (1<<15)`

NOTE_LENGTH	1u	1	255	The duration of each note in custom melody mode. *Units: 8ms samples*
DECAY_FACTOR	1u	0	15	Set the envelope "attack-decay" after each pause,that makes sound more natural. The bigger value, the longer decay. 0 - no decay. *Note: envelope takes effect only in the encoder-enabled firmware or when motors are OFF. The same is true for the 'volume' parameter in the GUI.

RESERVED	8b			
NOTE_FREQ_HZ[N]	2u*N	554	21000	Array of 2u elements, size N = 0..50, - melody to play if mode=BEEPER_MODE_CUSTOM_MELODY. Special value 21000 used to restart the envelope. Value > 21000 restarts envelope and makes a pause with the duration (val – 21000) 8ms-samples. *Units: Hz*

Example1: simple melody with short B5, D6, G6 notes and envelope:
```
00 80 05 03 00 00 00 00 00 00 00 00 DB 03 DB 03 08 52 DB 03 DB 03 08 52 96 04 96 04 08
52 1F 06 1F 06 1F 06 1F 06 1F 06
```

Example2: standard "calibration" sound:
```
01 00 00 03 00 00 00 00 00 00 00 00
```

Example3: single beep 1 second at 3kHz:
```
00 80 7D 00 00 00 00 00 00 00 00 00 B8 0B
```

CMD_SIGN_MESSAGE (#50) – sign message by secret keys

Name	Type	Min	Max	Possible values, remarks
SIGN_TYPE	1u			Defines a set of keys to be used
MESSAGE	32b			Message to be siged

Signed message is sent in response in the command CMD_SIGN_MESSAGE

CMD_EXT_IMU_CMD (#110) – forward message from the controller to the connected external IMU sensor

Name	Type	Min	Max	Possible values, remarks
CMD_ID	1u			Command ID (see GPS_IMU API specification for available commands)
DATA	...			Payload

The response from the external IMU will be sent back in the CMD_EXT_IMU_CMD incoming command.

CMD_EXT_SENS_CMD (#150) – forward message to the GPS_IMU sensor

(min. frw.ver. 2.68b7, “Extended” family only)

Forward message to the GPS_IMU sensor connected by CAN bus and acting as a main IMU.

Name	Type	Min	Max	Possible values, remarks
FLAGS	1u			Bit0: high priority. Low priority messages may be lost or delayed during the transmission, while delivery of high priority messages is guaranteed.
COMAND_ID	1u			Command ID according to GPS_IMU Serial API
DATA				All remaining bytes are sent as a payload. It doesn't include header and checksum.

All messages that GPS_IMU sends in response, are wrapped by CMD_EXT_SENS_CMD incoming message.

CMD_CAN_DEVICE_SCAN (#96) – scan for the connected CAN devices

No parameters.

Controller scans all connected CAN devices and answers with the CMD_CAN_DEVICE_SCAN incoming command.
CMD_ERROR is sent in case of problems, with the ERROR_CODE related to file operations.

CMD_TRANSPARENT_SAPI (#151) – send data to serial port on any device via CAN bus

(min. frw.ver. 2.72b0, "Extended" family only)

Command forwards data to a distant serial port on the device connected by CAN bus and supporting "transparent serial" functionality. Serial port on the target device should be enabled and configured for the function "*Transparent serial via Serial API*" in the GUI's "Service" tab. The serial data received on this serial ports' Rx will be forwarded via the same message to the device + port configured as "Target port"

> *NOTE: The outgoing data may be sent from any Serial API -enabled port, but the incoming data always goes to the target port specified in the configuration.*

Name	Type	Min	Max	Possible values, remarks
TARGET	1u			bits0..1 – port on the target device bits2..5 – target device (see definitions below) bit6 – wait flag: if set, in case TX buffer is full, blocks and wait. Otherwise, skips this data packet. bit7 – reserved `TARGET = PortId + (DeviceId<<2) + (WaitFlag<<6)`
PAYLOAD	...			All remaining bytes will be forwarded to the serial ports' Tx

Options for TARGET field and support of this function in devices:

Device	DeviceId	PortId	Transparent SAPI support
SBGC32	1	0: UART1 1: RC_SERIAL 2: UART2 3: USB_VCP / UART3	-
GPS_IMU	2	0: UART1 1: UART2	no*
CAN_IMU(main)	3	0: UART1 1: UART2	no*
CAN_IMU(frame)	4		
GPS_Split_Rcvr	5	0: UART1 1: UART2 2: UART3 3: EMBED_GNSS	no*
CAN_Serial_hub1	6	1: Serial1 2: Serial2 3: Serial3	no*
CAN_Serial_hub2	7		

CAN_Drv#1	8	0: UART2	CAN_DRV frw. 1.28 SBGC32 frw. 2.72b0
CAN_Drv#2	9		
CAN_Drv#3	10		
CAN_Drv#4	11		

** A support for these devices will be added later*

Responses

CMD_CONFIRM (#67) – confirmation of previous command or finished calibration

Name	Type	Min	Max	Possible values, remarks
CMD_ID	1u			Command ID to confirm
DATA	1u or 2u			DATA depends on command to be confirmed

CMD_ERROR (#255) – error executing previous command

Data depends on error type.

Name	Type	Min	Max	Possible values, remarks
ERROR_CODE	1u			Codes related to file operations: 0 - No error 1 - EEPROM access fault 2 - File is not found 3 - FAT records fault 4 - No free space left 5 - FAT is full 6 - File size is invalid 7 - CRC check failed 8 - Limit reached 9 - File corrupted 10 - Wrong params
ERROR_DATA	4b			

CMD_I2C_READ_REG_BUF (#40) – result of reading from I2C device

Name	Type	Min	Max	Possible values, remarks
DATA	1..255b			Data length depends on the DATA_LEN parameter in the request.

CMD_AUTO_PID (#35) – progress of PID auto tuning

This command is sent by the controller during the automatic PID tuning, if requested.

Name	Type	Min	Max	Possible values, remarks

P[3]		1u*3			
I[3]		1u*3			
D[3]		1u*3			
LPF_FREQ[3]		2u*3			
ITERATION_CNT		2u			
axis = (1..3)	TRACKING_ERROR	float			Current error between the target and actual system response
	RESERVED	6b			
RESERVED		10b			

CMD_RESET (#114) – notification on device reset

Device sent this command when goes to reset. There is a delay 1000ms after this command is sent and reset is actually done. External application can free up resources and properly close the serial connection.

MOTOR4_CONTROL - provides data for the external controller of the 4th axis motor

(frw.ver. 2.68+)

It's not a separate command. This structure is included as a part of other commands.

Name	Type	Min	Max	Possible values, remarks
FF_SPEED	2s			Feed-forward control *Units: 0,06103701895 degree/sec*
ANGLE_ERROR	2s			Distance to reach the target angle of 4th axis *Units: 0,02197265625 degree*
PID_OUT	4f			The output of the internal PID loop running over the ANGLE_ERROR with the FF_SPEED mixed, scaled by the 'scale factor' parameter.

CMD_EVENT (#102) – sent when event is triggered

(frw.ver. 2.65+)

Name	Type	Min	Max	Possible values, remarks
EVENT_ID	1u			**EVENT_ID_MENU_BUTTON = 1** generated on the menu buttons press, hold or release actions. For the "hold" state, command is sent serially with the given interval. Supported types: EVENT_TYPE_OFF, EVENT_TYPE_ON, EVENT_TYPE_HOLD **EVENT_ID_MOTOR_STATE = 2** generated on the motors ON/OFF action. Supported types: EVENT_TYPE_OFF, EVENT_TYPE_ON. **EVENT_ID_EMERGENCY_STOP = 3** generated on the emergency stop error. Supported types: EVENT_TYPE_OFF, EVENT_TYPE_ON **EVENT_ID_CAMERA = 4**

				generated on the menu commands "Camera Rec[Photo] event" Supported types: EVENT_TYPE_REC_PHOTO, EVENT_TYPE_PHOTO **EVENT_ID_SCRIPT = 5** *(frw. ver. 2.68b8+)* generated on script start (EVENT_TYPE_ON) and finish (EVENT_TYPE_OFF). PARAM1 holds the slot from where the script is executed.
EVENT_TYPE	1u			Possible value and its meaning depends on the EVENT_ID parameter. **EVENT_TYPE_OFF = 1** state changed to OFF (button is released, motor is turned OFF) **EVENT_TYPE_ON = 2** state is changed to ON (button is pressed, motors is turned ON) **EVENT_TYPE_HOLD = 4** state is remaining ON (button is held). **EVENT_TYPE_REC_PHOTO = 1** **EVENT_TYPE_PHOTO = 2** menu commands "Camera Rec/Photo event" and "Camera photo event"
PARAM1	2b			Possible value and its meaning depends on the EVENT_ID and EVENT_TYPE parameters: **EVENT_ID_MENU_BUTTON** for the "release" and "hold" events, encodes the time period when the button was held (unsigned value in milliseconds) **EVENT_ID_SCRIPT** slot from where the script is executed, starting from 0.
NOTE: this command may be expanded by extra parameters in future versions...				

CMD_SIGN_MESSAGE (#50) – result of message signing

Name	Type	Min	Max	Possible values, remarks
SIGNATURE	32b			Signed message

CMD_EXT_IMU_CMD (#110) – forwarded message received from the connected external IMU sensor

Name	Type	Min	Max	Possible values, remarks
CMD_ID	1u			Command ID (see GPS_IMU API specification for available commands)
DATA	...			Payload

CMD_EXT_SENS_CMD (#150) – forward message from the GPS_IMU sensor

(min. frw.ver. 2.68b7, "Extended" family only)

A forwarded message from the GPS_IMU sensor connected by CAN bus and acting as a main IMU. This message is sent in response to CMD_EXT_SENS_CMD outgoing message.

Name	Type	Min	Max	Possible values, remarks
COMAND_ID	1u			Command ID according to GPS_IMU Serial API
DATA				All remaining bytes are for payload. It doesn't include header and checksum.

CMD_CAN_DEVICE_SCAN (#96) – result of scanning all connected CAN devices, with the ID assigned to them.

Name		Type	Min	Max	Possible values, remarks
N=(1..DeviceNum)	UID	12b			Unique identifier of the device
	ID	1u			Assigned ID to this device, 0 if not assigned. 5 – CAN_IMU (main) 6 – CAN_IMU (frame) 7 – GPS_IMU (main) 17 – CAN_Drv#1 18 – CAN_Drv#2 19 – CAN_Drv#3 20 – CAN_Drv#4 21 – CAN_Drv#5 22 – CAN_Drv#6 23 – CAN_Drv#7 28 – CAN_IMU (main) (old) 29 – CAN_IMU (frame) (old) *On some firmware versions devices that can't be assigned, are not listed.*
	TYPE	1u			Bits 0..6: device type 1 – Motor driver 2 – IMU Bit7: ID is hardware-assigned

CMD_TRANSPARENT_SAPI (#151) – receive data from serial port on any device via CAN bus

(min. frw.ver. 2.72b0, "Extended" family only)

See the corresponding command definition CMD_TRANSPARENT_SAPI

EEPROM and internal file system

Requests

CMD_READ_FILE (#53) – read file from internal filesystem

This command reads a portion of data from a file with the identifier FILE_ID, started at PAGE_OFFSET pages (1page = 64byte) and to the end of file, but not more then MAX_SIZE bytes. Size of a portion should not exceed maximum allowed command data length (256 bytes). The result or error code is sent in the incoming command CMD_READ_FILE.

Name	Type	Min	Max	Possible values, remarks

FILE_ID	2u			1st byte encodes the file type; 2nd byte depends on type;
				`FILE_TYPE_SCRIPT = 1`
				`FILE_TYPE_IMU_CALIB = 3`
				`FILE_TYPE_COGGING_CORRECTION = 4`
				`FILE_TYPE_ADJ_VAR_LUT = 5`
				`FILE_TYPE_PROFILE_SET = 6`
				`FILE_TYPE_PARAMS = 7`
				`FILE_TYPE_TUNE = 8`
				`FILE_TYPE_CANDRV = 10`

PAGE_OFFSET	2u			offset from the beginning, in pages. 1 page = 64 bytes.
MAX_SIZE	2u			
RESERVED	14b			

CMD_WRITE_FILE (#54) – write file to internal filesystem

This command writes a portion of data to a file with the identifier FILE_ID. If file is not exists, it is created. If FILE_SIZE is not equal to existing file size, file is adjusted to new size. If DATA is empty, file is deleted.

Name	Type	Min	Max	Possible values, remarks
FILE_ID	2u			See CMD_READ_FILE.FILE_ID
FILE_SIZE	2u			Full size of a file
PAGE_OFFSET	2u			offset from the beginning, in pages. 1 page = 64 bytes.
DATA	?			All remaining bytes are counted as data. Size should be less then FILE_SIZE parameter. If data is empty, file will be deleted.

In response CMD_CONFIRM is sent, with parameter ERR_CODE. Possible codes:

```
NO_ERROR = 0
ERR_EEPROM_FAULT = 1
ERR_FILE_NOT_FOUND = 2
ERR_FAT = 3
ERR_NO_FREE_SPACE = 4
ERR_FAT_IS_FULL = 5
ERR_FILE_SIZE = 6
ERR_CRC = 7
ERR_LIMIT_REACHED = 8
ERR_FILE_CORRUPTED = 9
ERR_WRONG_PARAMS = 10
```

CMD_FS_CLEAR_ALL (#55) – delete all files from internal filesystem

Returns CMD_CONFIRM with parameter ERR_CODE (see definitions in the CMD_WRITE_FILE command)

CMD_EEPROM_WRITE (#47) – writes a block of data to EEPROM to specified address

Name	Type	Min	Max	Possible values, remarks
ADDR	4u	0	32767 *	address should be aligned to 64. *EEPROM size in all SBGC32 controllers is 32Kbytes.
DATA	?			All remaining bytes counted as data, arbitrary size but aligned to 64-byte pages

On success, confirmation CMD_CONFIRM is sent with parameters CMD_EEPROM_WRITE, ADDR.

CMD_READ_EXTERNAL_DATA (#42) – receive user data, stored in the EEPROM

External systems can use this area to store their configurations.

Name	Type	Min	Max	Possible values, remarks
DATA	128b			

CMD_EEPROM_READ (#48) – request a reading of block of data from EEPROM at the specified address and size.

Name	Type	Min	Max	Possible values, remarks
ADDR	4u	0	32767 *	address should be aligned to 64. *EEPROM size in all SBGC32 controllers is 32Kbytes.
SIZE	2u	64	192	size should be aligned to 64

On success, CMD_EEPROM_READ is sent in response.

CMD_WRITE_EXTERNAL_DATA (#41) – stores any user data to the dedicated area in the EEPROM

Name	Type	Min	Max	Possible values, remarks
DATA	128b			

Confirmation is sent on success.

CMD_READ_EXTERNAL_DATA (#42) – request user data, stored in the EEPROM

No parameters.

CMD_READ_EXTERNAL_DATA is sent in response.

Responses

CMD_READ_FILE (#53) – result of reading file from internal filesystem

In case of success:

Name	Type	Min	Max	Possible values, remarks
FILE_SIZE	2u			total size of file, bytes
PAGE_OFFSET	2u			offset that was requested, in pages. 1 page = 64 bytes
DATA	?			size that was requested, or less if the end of file is reached

In case of errors:

Name	Type	Min	Max	Possible values, remarks
ERR_CODE	1u			see error definitions in the CMD_WRITE_FILE command

CMD_EEPROM_READ (#48) – receive a portion of data read from EEPROM at the specified address.

Name	Type	Min	Max	Possible values, remarks
ADDR	4u			Address of a portion of data, 64-byte aligned
DATA	?			All remaining bytes are counted as data. Size is specified in the CMD_EEPROM_READ outgoing command.

Appendix

Command ID definitions

```
#define CMD_READ_PARAMS  82
#define CMD_WRITE_PARAMS  87
#define CMD_REALTIME_DATA  68
#define CMD_BOARD_INFO  86
#define CMD_CALIB_ACC  65
#define CMD_CALIB_GYRO  103
#define CMD_CALIB_EXT_GAIN  71
#define CMD_USE_DEFAULTS  70
#define CMD_CALIB_POLES  80
#define CMD_RESET  114
#define CMD_HELPER_DATA 72
#define CMD_CALIB_OFFSET  79
#define CMD_CALIB_BAT  66
#define CMD_MOTORS_ON  77
#define CMD_MOTORS_OFF  109
#define CMD_CONTROL  67
#define CMD_TRIGGER_PIN  84
#define CMD_EXECUTE_MENU 69
#define CMD_GET_ANGLES  73
#define CMD_CONFIRM  67
#define CMD_BOARD_INFO_3  20
#define CMD_READ_PARAMS_3 21
#define CMD_WRITE_PARAMS_3 22
#define CMD_REALTIME_DATA_3  23
#define CMD_REALTIME_DATA_4  25
#define CMD_SELECT_IMU_3 24
#define CMD_READ_PROFILE_NAMES 28
#define CMD_WRITE_PROFILE_NAMES 29
#define CMD_QUEUE_PARAMS_INFO_3 30
#define CMD_SET_ADJ_VARS_VAL 31
#define CMD_SAVE_PARAMS_3 32
#define CMD_READ_PARAMS_EXT 33
#define CMD_WRITE_PARAMS_EXT 34
#define CMD_AUTO_PID 35
#define CMD_SERVO_OUT 36
#define CMD_I2C_WRITE_REG_BUF 39
#define CMD_I2C_READ_REG_BUF 40
#define CMD_WRITE_EXTERNAL_DATA 41
#define CMD_READ_EXTERNAL_DATA 42
#define CMD_READ_ADJ_VARS_CFG 43
#define CMD_WRITE_ADJ_VARS_CFG 44
#define CMD_API_VIRT_CH_CONTROL 45
#define CMD_ADJ_VARS_STATE 46
#define CMD_EEPROM_WRITE 47
#define CMD_EEPROM_READ 48
#define CMD_CALIB_INFO 49
#define CMD_SIGN_MESSAGE 50
#define CMD_BOOT_MODE_3 51
#define CMD_SYSTEM_STATE 52
#define CMD_READ_FILE 53
#define CMD_WRITE_FILE 54
#define CMD_FS_CLEAR_ALL 55
```

```
#define CMD_AHRS_HELPER 56
#define CMD_RUN_SCRIPT 57
#define CMD_SCRIPT_DEBUG 58
#define CMD_CALIB_MAG 59
#define CMD_GET_ANGLES_EXT 61
#define CMD_READ_PARAMS_EXT2 62
#define CMD_WRITE_PARAMS_EXT2 63
#define CMD_GET_ADJ_VARS_VAL 64
#define CMD_CALIB_MOTOR_MAG_LINK 74
#define CMD_GYRO_CORRECTION 75
#define CMD_DATA_STREAM_INTERVAL 85
#define CMD_REALTIME_DATA_CUSTOM 88
#define CMD_BEEP_SOUND 89
#define CMD_ENCODERS_CALIB_OFFSET_4  26
#define CMD_ENCODERS_CALIB_FLD_OFFSET_4 27
#define CMD_CONTROL_CONFIG 90
#define CMD_CALIB_ORIENT_CORR 91
#define CMD_COGGING_CALIB_INFO 92
#define CMD_CALIB_COGGING 93
#define CMD_CALIB_ACC_EXT_REF 94
#define CMD_PROFILE_SET 95
#define CMD_CAN_DEVICE_SCAN 96
#define CMD_CAN_DRV_HARD_PARAMS 97
#define CMD_CAN_DRV_STATE 98
#define CMD_CAN_DRV_CALIBRATE 99
#define CMD_READ_RC_INPUTS 100
#define CMD_REALTIME_DATA_CAN_DRV 101
#define CMD_EVENT 102
#define CMD_READ_PARAMS_EXT3 104
#define CMD_WRITE_PARAMS_EXT3 105
#define CMD_EXT_IMU_DEBUG_INFO 106
#define CMD_SET_DEVICE_ADDR 107
#define CMD_AUTO_PID2 108
#define CMD_EXT_IMU_CMD 110
#define CMD_READ_STATE_VARS 111
#define CMD_WRITE_STATE_VARS 112
#define CMD_SERIAL_PROXY 113
#define CMD_IMU_ADVANCED_CALIB 115
#define CMD_API_VIRT_CH_HIGH_RES 116
#define CMD_CALIB_ENCODER_LUT 117
#define CMD_CALIB_ENCODER_LUT_RES 118
#define CMD_WRITE_PARAMS_SET 119
#define CMD_CALIB_CUR_SENS 120
#define CMD_CONTROL_EXT 121
#define CMD_ENC_INT_CALIB   122
#define CMD_SYNC_MOTORS   123
#define CMD_EXT_LICENSE_INFO 124
#define CMD_VIBRATION_TEST_START_STOP 125
#define CMD_VIBRATION_TEST_DATA 126
#define CMD_CAN_DRV_TELEMETRY 127

#define CMD_EXT_SENS_CMD 150
#define CMD_TRANSPARENT_SAPI 151

#define CMD_SET_DEBUG_PORT 249
#define CMD_MAVLINK_INFO 250
#define CMD_MAVLINK_DEBUG 251
```

```
#define CMD_DEBUG_VARS_INFO_3 253
#define CMD_DEBUG_VARS_3 254
#define CMD_ERROR  255
```

Appendix A: Examples and libraries

We provide a comprehensive C-language library with examples for various platforms:
https://github.com/basecamelectronics/sbgc32-serial-api

See README.md for details.

CRC16 reference implementation in C

```
void crc16_update(uint16_t length, uint8_t *data, uint8_t crc[2]) {
    uint16_t counter;
    uint16_t polynom = 0x8005;
    uint16_t crc_register = (uint16_t)crc[0] | ((uint16_t)crc[1] << 8);
    uint8_t shift_register;
    uint8_t data_bit, crc_bit;

    for (counter = 0; counter < length; counter++) {
        for (shift_register = 0x01; shift_register > 0x00; shift_register <<= 1) {
            data_bit = (data[counter] & shift_register) ? 1 : 0;
            crc_bit = crc_register >> 15;
            crc_register <<= 1;

            if (data_bit != crc_bit) crc_register ^= polynom;
        }
    }

    crc[0] = crc_register;
    crc[1] = (crc_register >> 8);
}

void crc16_calculate(uint16_t length, uint8_t *data, uint8_t crc[2]) {
    crc[0] = 0; crc[1] = 0;
    crc16_update(length, data, crc);
}
```

Example with CRC16 for command `CMD_BOARD_INFO`:

start byte	header			payload		CRC16	
	command ID	payload size	header checksum				
0x24	0x56	0x02	0x58	0x00	0x00	**0xE6**	**0x13**

Appendix B: Run-time parameters definition (adjustable variables)

Used with CMD_SET_ADJ_VARS_VAL (#31)

NAME	*Frw. ver.*	ID	TYPE	MIN	MAX	REMARK
P_ROLL P_PITCH P_YAW		0 1 2	1u	0	255	
I_ROLL I_PITCH I_YAW		3 4 5	1u	0	255	
D_ROLL D_PITCH D_YAW		6 7 8	1u	0	255	
POWER_ROLL POWER_PITCH POWER_YAW		9 10 11	1u	0	255	
ACC_LIMITER		12	2s	0	1275	Units: degrees/sec^2
FOLLOW_SPEED_ROLL FOLLOW_SPEED_PITCH FOLLOW_SPEED_YAW		13 14 15	1u	0	255	
FOLLOW_LPF_ROLL FOLLOW_LPF_PITCH FOLLOW_LPF_YAW		16 17 18	1u	0	15	
RC_SPEED_ROLL RC_SPEED_PITCH RC_SPEED_YAW		19 20 21	1u	0	255	
RC_LPF_ROLL RC_LPF_PITCH RC_LPF_YAW		22 23 24	1u	0	15 (255)*	**Range depends on the flag "Extend LPF range" in GUI settings*
RC_TRIM_ROLL RC_TRIM_PITCH RC_TRIM_YAW		25 26 27	1s	-127	127	
RC_DEADBAND		28	1u	0	255	Updates RC Dead-band for all axes. *Frw.2.72b0: added new variables for PITCH, YAW axes; once they are updated, this variable affects only the ROLL axis.*
RC_EXPO_RATE		29	1u	0	100	Updates RC Expo rate for all axes. *Frw.2.72b0: added new variables for PITCH, YAW axes; once they are updated, this variable affects only the ROLL axis.*
FOLLOW_PITCH		30	1u	0	1	
FOLLOW_YAW_PITCH		31	1u	0	2	0 – disabled 1 - Follow YAW 2* – Follow YAW, PITCH [ROLL] **frw. ver. 2.65b3*
FOLLOW_DEADBAND		32	1u	0	255	
FOLLOW_EXPO_RATE		33	1u	0	100	

FOLLOW_ROLL_MIX_START		34	1u	0	90	
FOLLOW_ROLL_MIX_RANGE		35	1u	0	90	
GYRO_TRUST		36	1u	0	255	Special value 0 disables accelerometer completely
FRAME_HEADING_ANGLE		37	2s	-1800	1800	The frame's heading (YAW) angle expressed in Euler order frame-PITCH-ROLL-YAW used as an absolute heading reference for the gyrosope sensor. Effect is similar to the FRAME_HEADING variable in the CMD_HELPER_DATA. *Units: 0.1 degrees* *Special value 0x7FFF disables the correction*
GYRO_HEADING_CORRECTION		38	2s	-20000	20000	Units: 0.001 of gyro sensor units
ACC_LIMITER_ROLL ACC_LIMITER_PITCH ACC_LIMITER_YAW		39 40 41	2s	0	1275	Units: degrees/sec^2
PID_GAIN_ROLL PID_GAIN_PITCH PID_GAIN_YAW		42 43 44	1u	0	255	Gain is calculated as 0.1 + PID_GAIN[axis]*0.02
LPF_FREQ_ROLL LPF_FREQ_PITCH LPF_FREQ_YAW		45 46 47	2u	10	400	Units: Hz
TIMELAPSE_TIME		48	2u	1	3600	Units: sec
MAV_CTRL_MODE		49	1u	0	2	0 – disabled 1 – ROLL and PITCH only 2 – enabled for all axes
H_CORR_FACTOR	*2.68b7*	50	1u	0	255	Heading correction factor from external reference
SW_LIM_MIN.ROLL SW_LIM_MAX.ROLL SW_LIM_MIN.PITCH SW_LIM_MAX.PITCH SW_LIM_MIN.YAW SW_LIM_MAX.YAW	*2.68b8*	51 52 53 54 55 56	2s	-3600	3600	Software limits for each motor, degrees (encoder firmware only) *Note: set new values only in pairs, min should go prior to max!*
FOLLOW_RANGE.ROLL FOLLOW_RANGE.PITCH FOLLOW_RANGE.YAW	*2.68b9*	57 58 59	1u	0	255	Units: degrees
AUTO_PID_TARGET	*2.68b9*	60	1u	0	255	Stability-precision slider for automatic PID tuing algorithm
RC_MODE.ROLL RC_MODE.PITCH RC_MODE.YAW	*2.69b3*	61 62 63	1u			0 – ANGLE 1 – SPEED 2 – TRACKING
EULER_ORDER	*2.69b3*	64	1u			0 – PITCH-ROLL-YAW 1 – ROLL-PITCH-YAW 2 – PITCH(M)-ROLL-YAW(M) 3 – ROLL-PITCH(M)-YAW(M) 4 – YAW-ROLL-PITCH
FOLLOW_IN_DBAND	*2.70b4*	65	1u	0	255	“Follow inside deadband” parameter
RC_LIMIT_MIN.ROLL RC_LIMIT_MAX.ROLL	*2.72b0*	66 67	2s	-3600	3600	Angle limits for the Euler axes, degrees.

RC_LIMIT_MIN.PITCH RC_LIMIT_MAX.PITCH RC_LIMIT_MIN.YAW RC_LIMIT_MAX.YAW		68 69 70 71				*Note: set new values only in pairs, min should go prior to max!*
RC_DEADBAND.PITCH RC_DEADBAND.YAW	*2.72b0*	72 73	1u	0	255	See RC_DEADBAND
RC_EXPO_RATE.PITCH RC_EXPO_RATE.YAW	*2.72b0*	74 75	1u	0	100	See RC_EXPO
SHAKE_AMP.ROLL SHAKE_AMP.TILT SHAKE_AMP.PAN	*2.73.x*	76 77 78	1u	0	255	See SHAKE_GENERATOR_CFG specification
SHAKE_FREQ		79	1u	0	255	
SHAKE_FREQ_RANGE		80	1u	0	100	
SHAKE_PAUSE_PERIOD		81	2u	500	5600	
SHAKE_PAUSE_BALANCE		82	1u			
SHAKE_PAUSE_RANDOMNESS		83	1u	0	100	
SHAKE_RESONANCE_GAIN.ROLL		84	1u	0	255	
SHAKE_RESONANCE_GAIN.TILT		85	1u	0	255	
SHAKE_RESONANCE_GAIN.PAN		86	1u	0	255	
SHAKE_FREQ_SHIFT.ROLL		87	1s	-127	127	
SHAKE_FREQ_SHIFT.TILT		88	1s	-127	127	
SHAKE_FREQ_SHIFT.PAN		89	1s	-127	127	

Appendix C: Providing external reference attitude/heading information from UAV

Serial API allows for flight controllers of UAVs to send attitude and heading information that can be used as a reference to correct attitude and heading of internal IMU, improving its precision. As a rule, flight controllers have more sensors on-board and can do better attitude/heading angles estimation than the IMU sensor used in the SBGC32 controller.

This kind of correction is described in detail in section 18 of the "SimpleBGC32 User Manual". Our controller supports direct connection only for several models of AHRS/IMU devices. For others, Serial API can be used.

In a few words, there are two options to apply the correction:

1. provide attitude and/or heading of the frame via command `CMD_AHRS_HELPER`
2. compensate for linear accelerations via command `CMD_HELPER_DATA`

Option 1) is better because the attitude/heading information is used directly, allowing to disable the internal accelerometer and keep using the internal gyroscope only. However, it requires knowing an exact attitude of the gimbal's frame. But a common case when the gimbal mounted on the UAV has anti-vibration dampeners, which add some degree of freedom. It makes using attitude/heading information from the UAV's flight controller not applicable for a precise attitude correction for the gimbal's IMU.

Option 2) is more tolerant in this case. You can use it to compensate the attitude drift caused by an accelerated motion and the heading drift causing by an unreferenced YAW gyroscope. Attitude/heading is still computed inside the gimbal's controller, using an internal accelerometer and gyroscope. You need to pass linear accelerations (with the gravity subtracted) in the command `CMD_HELPER_DATA`. Also, for 3-axis systems, you have to provide heading information to synchronize the camera's heading angle with the UAV's heading. “Heading” here is the Euler's YAW angle expressed in order “frame-PITCH-ROLL-YAW” (note that the order of angles does matter; the commonly used in aeronautics “ROLL-PITCH-YAW” will give different values for the same physical orientation).

As a drawback, option 2) does not help to compensate for a drift caused by the thermal instability of the gyroscope and accelerometer sensors.

Using high-grade IMU for a correction

If a high-grade IMU is used instead of UAV, there are several options where to mount it: on the frame (above the outer motor), below the outer motor, or on the camera platform. The last option provides the best accuracy because mechanical imperfection and encoder calibration do not distort the AHRS data.

Notes on data rates and how to interrupt the correction

The more data rate is, the better. Though, as this correction is a kind of low-pass filter, and it changes the actual attitude very slowly, it is okay to have a slow data rate and still have a good result. There are no reasons to have it higher than 125 Hz as it's the maximum processing rate of Serial API messages.

The last data received in the `CMD_AHRS_HELPER / CMD_HELPER_DATA` messages is considered as the actual attitude/heading until it's updated by the new portion of data. If the host controller stops sending these messages, but system actually moves, it leads to serious confusion of the IMU subsystem.
Starting from firmware ver. `2.70b1` it is possible to properly stop the correction by sending a command `CMD_AHRS_HELPER` with the flag "Disable correction" (bit10 in the `MODE` parameter). For the `CMD_HELPER_DATA` it's enough to send zero accelerations and special value `32767` for the `FRAME_HEADING` parameter.

Appendix D: Coordinate system conversions

If not specified, it's assumed the END (East-North-Down) coordinate system, that differs from commonly used NED system: in our system, X points right (or East), Y points forward (or North), Z points down. To convert vectors to NED system, you need to swap X and Y components.

Rotation matrix

CMD_AHRS_HEPER provides and takes orientation data in a form of rotation matrix (DCM), but first row is omitted to save the bandwidth. The full rotation matrix can be reconstructed from two vectors H_VECT and Z_VECT:

$$DCM_{END} = \begin{pmatrix} A_x & A_y & A_z \\ H_x & H_y & H_z \\ Z_x & Z_y & Z_z \end{pmatrix} = \begin{pmatrix} A_{VECT} \\ H_{VECT} \\ Z_{VECT} \end{pmatrix},$$

$$A_{VECT} = H_{VECT} \times Z_{VECT}$$

Z_VECT and H_VECT can be considered as unit vectors in body reference system pointing down and North directions.

To convert rotation matrix from / to commonly used NED coordinate system, just swap first two rows:

$$DCM_{NED} = \begin{pmatrix} H_x & H_y & H_z \\ A_x & A_y & A_z \\ Z_x & Z_y & Z_x \end{pmatrix}$$

Note that some systems need an alternative definition of rotation matrix: (body-to-world or world-to-body defines the same rotations but in opposite direction), so DCM may need to be transposed (i.e. rows arranged as columns).

Quaternions

Serial API does not support quaternions prior to firmware version 2.73.x, but the rotation matrix can be converted to / from quaternion without loss of precision.

Coordinate system conversion:

- In order to convert quaternion [w, x,y,z] between RFU (ENU) and our RFD (END) systems, change the sign of x and y.
- To convert between FRD (NED) and our RFD (END), swap x and y and change the sign of x, y, z.

±180/360° rotation ambiguity

Any orientation can be encoded by two quartenions having an opposite direction. In commands we use full scale for quartenions (angle is in ±360°range in axis-angle meaning), unless FLAG_ATTITUDE_LIMITED_180 is set. When converting from other formats, take care about range and direction. For example, assuming the last TARGET_ATTITUDE was 179° over the Z axis, in order to make 2-degree step, the next command should have angle = 181° rather than -179°.

Euler angles

Serial API provides and takes all angles in a form of *Euler angles*. Attitude/heading can be reconstructed from Euler angles, but it is important taking into account the order of Euler rotations. SBGC32 supports several orders that may be configured in system parameters. Since this order is not knows in advance, it's better to avoid using Euler angles to represent the rotations in calculations. Another problem is that Euler rotations are not free from *gimbal lock* problem. Rotation matricies or quaternions are free from this problem and correctly represents all possible positions.

Serial API uses the following convention for Euler angles:

- ROLL is over Y axis (forward), clockwise positive (if looking towards the axis direction)
- PITCH is over X axis (right), counter-clockwise positive
- YAW is over Z axis (down), clockwise positive

If converted from/to the commonly used convention, the PITCH angle should be inverted.

Appendix E: "Emergency stop" error codes

- SUB_ERR_I2C_ERRORS = 1, // High rate of I2C errors
- SUB_ERR_DRV_OTW = 2, // Driver over-temperature protection
- SUB_ERR_DRV_FAULT = 3, // Driver fault (under-voltage, over-current, short circuit)
- SUB_ERR_ENCODER_IMU_ANGLE = 4, // Encoder/IMU angles mismatch
- SUB_ERR_CALIBRATION_FAILED = 5, // Auto calibration process caused serious fault
- SUB_ERR_INTERNAL_SYSTEM_ERROR = 6, // Stack is damaged
- SUB_ERR_ENCODER_CALIB_BAD_SCALE = 7, // estimated scale differs a lot from configured
- SUB_ERR_OVER_TEMPERATURE = 8, // MCU or power board over temperature
- SUB_ERR_BAD_MOTOR_POLES_INVERT = 9, // motor n.poles or inversion is wrong
- SUB_ERR_NOT_ENOUGH_MEMORY = 10, // static_malloc() can't allocate memory
- SUB_ERR_IMU_SENSOR_NOT_RESPONDING = 11, // lost connection to IMU sensor
- SUB_ERR_CAN_HARD = 12, // CAN on board hardware error
- SUB_ERR_MOTOR_OVERHEAT_PROTECTION = 13, // overheat protection is triggered
- SUB_ERR_MOTOR_IS_LOCKED = 14, // motor is locked during automated task
- SUB_ERR_BAD_IMU_HEALTH = 15, // IMU gyroscope and accelerometer error is too big: sensor sends corrupted data or wrong use conditions
- SUB_ERR_INFINITE_RESET = 16, // Infinite reset loop is detected
- SUB_ERR_WRONG_INITIAL_POSITION = 17, // wrong position: failed to detect encoder angle, or angle is outside soft limits
  SUB_ERR_MOTOR_LOAD_TIME_EXCEEDED = 18, // motors are fully loaded too long time
- SUB_ERR_CAN_DRV_OVERCURRENT = 19, // hardware short-circuit protection
- SUB_ERR_CAN_DRV_UNDERVOLTAGE = 20, // hardware or software undervoltage protection
- SUB_ERR_CAN_DRV_EMERGENCY_PIN = 21, // external emergency is triggered
- SUB_ERR_CAN_DRV_FOC_DURATION = 22, // FOC algorithm duration error
- SUB_ERR_CAN_DRV_MCU_OVERHEAT = 23, // driver temperature is to high
- SUB_ERR_CAN_DRV_MOTOR_OVERHEAT = 24, // motor temperature is to high
- SUB_ERR_CAN_DRV_OVERCURRENT_SOFT = 25, // current through motor exceed limit
- SUB_ERR_CAN_DRV_SEVERAL = 26, //several errors on driver
- SUB_ERR_CAN_EXT_BUS_OFF = 27, // CAN bus high rate errors of slave controller
- SUB_ERR_CAN_INT_BUS_OFF = 28, // CAN bus high rate errors of main controller
- SUB_ERR_ENCODER_NOT_FOUND = 29, // no any answer from encoder during init
- SUB_ERR_CAN_DRV_NOT_RESPONDING = 30, // lost connection to CAN Drv
- SUB_ERR_CAN_DRV_WRONG_PARAMS = 31, // some params of CAN Drv isn't correct
- SUB_ERR_OVERCURRENT = 32, // fast over current protection of main controller, or short circuit detection on startup
- SUB_ERR_UNSAFE_VOLTAGE = 33, // Under voltage protection or supply protection controller fault
- SUB_ERR_WRONG_FULL_BAT_VOLTAGE_PARAM = 34, //battery voltage is higher than expected at startup sequence
- SUB_ERR_EEPROM_PARAMS_CORRUPTED = 35, // parameters are corrupted in EEPROM and can't be restored from backup slot
- SUB_ERR_ENCODER_UNSUPPORTED_TYPE = 36, // unsupported type of encoder
- SUB_ERR_EXT_IMU_UNSUPPORTED_TYPE = 37, // unsupported type of external imu of CAN Imu, need update CAN Imu FW
- SUB_ERR_EXT_IMU_SENSOR_NOT_RESPONDING = 38, // lost connection to external imu of CAN Imu
- SUB_ERR_EXT_IMU_WRONG_PARAMS = 39, // some errors in self test in external imu of CAN Imu, or difference of int. and ext. is too big
- SUB_ERR_DRIVER_INIT = 40, // initialization of ext. motor driver failed
- SUB_ERR_EEPROM_VARS_OUT_OF_BORDERS = 41, // _write_vars() or _read_vars() out of borders
- SUB_ERR_IWDG_RESET = 42, // system was reset by watch-dog timer

- SUB_ERR_ADC_WAIT = 43, // error waiting for ADC samples
- SUB_ERR_CALIB_MOTOR_OFF = 44, // need power motor for calibration process
- SUB_ERR_TIMEOUT = 45,
- SUB_ERR_CAN_DRV_CALIB_UNEXPECTED_RES = 46, // CAN DRV RL calibration error
- SUB_ERR_MAG_SENSOR_NOT_RESPONDING = 47, // lost connection to MAG sensor
- SUB_ERR_CAN_DRV_OVERSPEED = 48, // CAN Drv spin too fast
- SUB_ERR_ENC_SELF_CALIB_FAILED = 49, // encoder self calibration failed
- SUB_ERR_CAN_VERSION_MISMATCH = 50, // major version of CAN module don't match SBGC firmware version
- SUB_ERR_CAN_SERIAL_INIT = 51, // Serial-over-CAN can't be configured with the current parameters
- SUB_ERR_CAN_SERIAL_CONFLICT = 52, // Serial-over-CAN conflict
- SUB_ERR_CAN_DRV_INIT_STAGE = 53, // CAN Drv initialization sequence failed though it responds
- SUB_ERR_ENCODER_DATA_TIMEOUT = 54, // no fresh data from encoder for a long time
- SUB_ERR_HSE_START_FAIL = 56, // HSE (quartz) start problem
- SUB_ERR_FRAME_IMU_SENSOR_NOT_RESPONDING = 57, // lost connection to FRAME IMU sensor
- SUB_ERR_PARAMETERS_MODIFIED = 58, // can't write parameters - they were modified in runtime
- SUB_ERR_NOT_NORMAL_POSITION = 59, // can do certain calibrations only in a normal position (frame is inverted)
- SUB_ERR_CAN_DRV_LIMIT_EXCEED = 60, // CAN Drv modules number exceed license

Appendix F: Compressed quaternion format

The compressed quaternion format takes 8 bytes instead of 16 bytes required to store it in 4 floats [w, x, y, z], whilst preserving high enough precision.

C-style structure definition with bit fields:

```
const float SCALE_FACTOR = 741453.7859759028838510909761497 3f;

struct {
      // component a
      uint32_t a : 19;
      // component a sign (1 - negative)
      uint16_t a_sign : 1;
      // component b
      uint32_t b : 19;
      // component b sign
      uint16_t b_sign : 1;
      // component c
      uint32_t c : 19;
      // component c sign
      uint16_t c_sign : 1;
      // index of the largest component in quaternion, 0..3
      uint16_t largest : 2;
      // sign of the largest component (1 - negative).
      uint16_t largest_sign : 1;
      // not used, padding to 64 bits
      uint16_t reserved : 1;
} quat_packed_t;
```

Compressing to packed format:

1. Find the largest component and store its index (0..3) in the 'largest' field. If the value is negative, set the 'largest_sign' bit to 1.
2. Take an absolute value of other 3 components, multiply them by `SCALE_FACTOR` and store to a, b, c fields (preserving the original order). If any value is negative, set the corresponding sign bit to 1.

Restoring from packed format:

1. Restore 3 components a, b, c to floats by multiplying by `1/SCALE_FACTOR` and taking into account the 'sign' bit.
2. Restore the 4th component referenced by the 'largest' index, as `sqrt(1 – (a*a + b*b + c*c))`, taking into account the 'largest_sign' bit.