Vertiq Standard DroneCAN (aka UAVCANv0) Support

This document details the standard DroneCAN messages supported across all Vertiq DroneCAN modules currently. The structure and contents of these messages are defined by the DroneCAN specification, this document just provides details on how exactly Vertiq modules support them and some supplementary information.

Details on the DroneCAN protocol can be found here on the DroneCAN specification. For a full listing of all standard messages specified by DroneCAN, see the List of Standard Data Types in the DroneCAN specification.

1 Broadcast Messages **1**
- 1.1 uavcan.protocol.NodeStatus (Data Type ID = 341) 2
- 1.2 uavcan.equipment.esc.Status (Data Type ID = 1034) 2
- 1.3 uavcan.equipment.device.Temperature (Data Type ID = 1110) 2
- 1.4 uavcan.equipment.esc.RawCommand (Data Type ID = 1030) 3

2 Service Requests **3**
- 2.1 uavcan.protocol.GetNodeInfo (Data Type ID = 1) 3
- 2.2 uavcan.protocol.param.GetSet (Data Type ID = 11) 3
- 2.3 uavcan.protocol.RestartNode (Data Type ID = 5) 4
- 2.4 uavcan.protocol.file.BeginFirmwareUpdate (Data Type ID = 40) 4

3 Configuration Parameters **4**
- 3.1 Node ID 5
- 3.2 Bitrate 5
- 3.3 ESC Index 5
- 3.4 Zero Behavior 5
- 3.5 Telemetry Frequency 6

1 Broadcast Messages

These are broadcast messages that are sent to or from the motors. Since they are broadcast, there is no response message sent. These messages typically make up the majority of communication on the bus during operation.

1.1 uavcan.protocol.NodeStatus (Data Type ID = 341)

The DroneCAN protocol requires that all nodes on a DroneCAN network periodically publish their status using the Node Status message. Vertiq modules support this behavior to conform to the standard. A *uavcan.protocol.NodeStatus* message is published at 1 Hz during operation, reporting its health, current mode, and uptime.

In normal operation a Vertiq module should always report its Health as OK, and its mode as Operational. No sub-modes or vendor specific status codes are currently supported.

Refer to the relevant section of Standard Data Types in the specification for more details

1.2 uavcan.equipment.esc.Status (Data Type ID = 1034)

This message is published periodically and provides telemetry updates on the state of the motor and its inputs. Specifically it contains information on:

- **Error Count:** This is a counter of CAN bus errors, specifically it details the number of transmit errors the motor has encountered.
- **Voltage:** The input voltage to the motor in volts
- **Current:** The current draw of the motor in amps
- **Temperature:** The temperature of the motors coils in Kelvin
- **RPM:** The current speed of the motor in RPM
- **Power Rating Percentage:** The PWM duty cycle percentage being applied to the motor, from 0% to 100%. Maximum power draw occurs at 100% duty cycle
- **ESC Index:** The ESC index of the motor that is broadcasting this update

The frequency that this message is published at is determined by the Telemetry Frequency configuration parameter.

Refer to the relevant section of Standard Data Types in the specification for more details

1.3 uavcan.equipment.device.Temperature (Data Type ID = 1110)

This message is published periodically and provides updates on the temperature of the microcontroller used on the controller. The fields contained in this message are:

- **Device ID:** The ESC index of the motor sending this broadcast
- **Temperature:** The temperature of the microcontroller in Kelvin. Note that this is different from the temperature in the ESC status message, as that is the temperature of the coils.
- **Error Flags:** Indicates if the motor is overheating

The frequency that this message is published at is determined by the Telemetry Frequency configuration parameter.

Refer to the relevant section of Standard Data Types in the specification for more details

1.4 uavcan.equipment.esc.RawCommand (Data Type ID = 1030)

Used to control the speed and direction of the motor. This message should be sent from the flight controller to the motors. The payload consists of a list of up to 20 values, with each value interpreted as a 14 bit signed integer. Each entry in the list corresponds to a motor with the given ESC index. E.g. The first entry in the list of raw commands will be read by the motor that has been assigned ESC index 0, the second entry will be read by the motor with ESC index 1, and so on.

The values represent the speed and direction to command the motor to, normalized over a range of [-8192, 8191], with -8192 being full speed in reverse and 8191 being full speed forward.

Refer to the relevant section of Standard Data Types in the specification for more details

2 Service Requests

Service requests are messages sent to a specific target node from another node, and to which the sending node expects to receive a response message.

2.1 uavcan.protocol.GetNodeInfo (Data Type ID = 1)

This request has no payload fields. The response message from the receiving node should include the status of the node as defined by the NodeStatus message, the software and hardware version of the node, and the name of the node. For Vertiq modules, the name of each node is currently “iq_motion.esc”.

Refer to the relevant section of Standard Data Types in the specification for more details

2.2 uavcan.protocol.param.GetSet (Data Type ID = 11)

Used to get or set the value of a configuration parameter using either the index or the name of the parameter. The configuration parameters currently available on standard Vertiq modules are listed in the Configuration Parameters section below. The request message should contain the index of the parameter or the name of the parameter depending on how you are accessing it, and it should contain a value to set the parameter to if you are setting it. **Parameter indices are not guaranteed to remain consistent across firmware versions. Indices should only be used for parameter discovery, when accessing the parameter directly it is recommended to always use the name**

The response message will contain the value of the parameter, information about its default, minimum, and maximum values, and the name of the parameter.

Refer to the relevant section of Standard Data Types in the specification for more details

2.3 uavcan.protocol.RestartNode (Data Type ID = 5)

This request has one field, its “magic number”. This field is an unsigned 40 bit integer. The magic number is used to identify that this is an intentional restart request, and we have also extended it to allow for two different types of reboots. One magic number performs a normal reboot that simply restarts into the normal application firmware, and the other reboots the device into the ST bootloader, which allows it to be programmed with new firmware. The reboot-into-bootloader can be useful for firmware updates. The numbers for each are:

- Normal Reboot Magic Number = 0xACCE551B1E
- Reboot to Bootloader Magic Number = 0xBEEFCAFE

The response to this request contains one boolean field. This field will be true if the number received was a valid magic number and the motor is restarting, and false if the number received was not one of the valid numbers.

Refer to the relevant section of Standard Data Types in the specification for more details.

2.4 uavcan.protocol.file.BeginFirmwareUpdate (Data Type ID = 40)

On modules that support DroneCAN firmware updates, this command causes the module to enter into upgrade mode, and begin attempting to firmware update over DroneCAN. The request includes a source node ID and a file path on that source node. The module will attempt to get information on the file from the file server, and then begin reading the data from the upgrade file.

On modules that do not support DroneCAN firmware updates, the module will respond with an INVALID_MODE error code to this request.

Currently, the Vertiq 81-08 does not support DroneCAN firmware updates.

Refer to the relevant section of Standard Data Types in the specification for more details.

3 Configuration Parameters

This is a listing of the configuration parameters that standard Vertiq modules support. Configuration parameters can be accessed using *uavcan.protocol.param.GetSet* requests.**The IDs of parameters are not guaranteed to remain consistent across firmware versions. IDs should only be used for parameter discovery, when accessing the parameter always use the name.**

Not all parameters listed here are supported by all modules currently. The parameters that are available on any given module can be discovered by iterating through the parameter indices with a *uavcan.protocol.param.GetSet* request until an empty response is returned.

Note that configuration parameter names are generally kept to 16 characters or less, because MAVLINK cannot handle parameter names greater than 16 characters. This simplifies integration with flight controllers that use MAVLINK.

3.1 Node ID

Name = “uavcan_node_id”
Type = Integer

This parameter defines the Node ID of the module on the UAVCAN network. This ID is how the node identifies itself when sending and receiving messages. No two nodes should have the same Node ID. ID 0 is typically reserved for the flight controller. A reboot is typically required after changing this parameter for the device to use the new Node ID on the network.

This parameter can also be changed through the IQ Control Center if you wish to change this without using DroneCAN.

3.2 Bitrate

Name = “bit_rate”
Type = Integer

This parameter determines the DroneCAN bitrate that the module will use in bit/s. This parameter takes effect immediately when changed, so if this is changed it will most likely be necessary to reconnect to the bus as at the new bitrate to continue communicating with the module. Note that most Vertiq modules do not currently support this parameter, support will be added in future releases. If this parameter is not supported, the module’s bitrate is fixed at 500000 bit/s.

3.3 ESC Index

Name = “esc_index”
Type = Integer

This parameter defines the ESC index of the module. The ESC index is used when a Raw Command message is received to determine which value in the Raw Command should be read by the module.

This parameter can also be changed through the IQ Control Center if you wish to change this without using DroneCAN.

3.4 Zero Behavior

Name = “zero_behavior”
Type = Integer

This parameter defines the behavior of the motor when it receives a zero setpoint in a Raw Command message. There are three possible behaviors: Coast, Brake, and Normal Control;er. By changing the integer value of Zero Behavior, the user can determine which of these behaviors is used. Descriptions of each behavior and the integer value of the Zero Behavior parameter corresponding to them are given below:

- **Coast (Value = 0):** The module will stop trying to drive the motor, releasing control and letting it simply coast to a stop.
- **Brake (Value = 1):** The motor will brake to stop itself, so the motor will stop suddenly on a zero setpoint instead of coasting gently to a stop. After stopping, it will then coast.
- **Normal Controller (Value = 2):** The normal controller continues to run, and will try to drive the motor at 0% throttle. This can cause vibrating or twitching in velocity mode if the PID gains are high.

This parameter can also be changed through the IQ Control Center if you wish to change this without using DroneCAN.

3.5 Telemetry Frequency

Name = “telem_frequency”
Type = Integer

This parameter defines the frequency in Hz with which the telemetry messages (uavcan.equipment.esc.Status and uavcan.equipment.device.Temperature) are broadcast by the module. For example, if this value were set to 10, the telemetry would be sent at a rate of 10 Hz. If this parameter is set to 0, no telemetry messages will be sent.

This parameter can also be changed through the IQ Control Center if you wish to change this without using DroneCAN.