

TM-UAVCAN-V2.3

1 Scope and Version Definition

1: This document primarily describes the communication protocol between the Electronic Speed Controller (ESC) and external devices.

2: The CAN protocol complies with the UAVCAN/DRONECAN protocol specifications. For more details on the protocol and to access related examples, please visit <https://github.com/dronecan/libcanard>.

2 Terminology

Table2-1 Document Terminology

Terminology	Explanation and Description
ESC	Electric Speed Controller ESC
CAN	Controller Area Network a communication protocol

3 Reference Documents

《CANBus Specifications v2.0》

4 Protocol Definition

4.1 TM-UAVCAN protocol

TM-UAVCAN protocol is based on the standard CANBus 2.0B protocol, utilizing 29-bit extended frame data frames.

TM-UAVCAN Hardware Layer Configuration	
CAN_SJW	CAN_SJW_1tq
CAN_BS1	CAN_BS1_4tq
CAN_BS2	CAN_BS2_1tq
SamplePoint	83.3%
BusClodError	0%

4.1.1 Concepts

1: Message Frame - It is a broadcast frame, and all nodes can receive this message.

2: Service Frame - It is a non-broadcast frame, with a specified node ID, where the node requesting the service expects a service response. Note: All transmission frames include both single-frame and multi-frame formats.

4.1.2 ID field

In the TM-UAVCAN protocol, we use only the data frame type from the five frame types defined in CANBus 2.0B. All data is transmitted via data frames. We define the arbitration segment ID code of the data frame in the following format:

Message frame

Field name	Priority					Message type ID																	Service not message								
																							Source node ID								
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Allowed values																							0	1...127							
CAN ID bytes	3					2					1							0													

Anonymous message frame

Field name	Priority					Discriminator										Lower bits of message type ID			Service not message										
																			Source node ID										
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																			0										
CAN ID bytes	3					2					1					0													

Service frame

Field name	Priority					Service type ID										Request not response								Service not message								
																Destination node ID								Source node ID								
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Allowed values																1...127								1	1...127							
CAN ID bytes	3					2										1								0								

For ESC's CAN bus communication application, only the Message frame and Service frame are currently used. Therefore, the following text will not cover the explanation of the Anonymous Message frame.

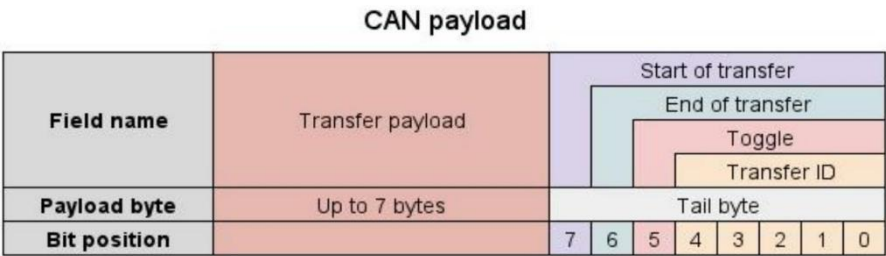
Bit Definition and Terminology Explanation

Bit Definition	Explanation and Description		
Priority	1:Priority indicates the priority of the CAN data frame. 2:The priority value range is 0 to 31. 3:The highest priority is 0, and the lowest priority is 31. 4: The TM-UAVCAN protocol defines priorities as shown in the table below:	Priority	Value
		HIGHEST	0x00
		HIGH	0x08
		MEDIUM	0x10
		LOW	0x18
		LOWEST	0x1F
Message type ID	The range of the Message Type ID is from 0 to 65535, inclusive of both 0 and 65535.		
Service type ID	The range of the Service Type ID is from 0 to 255, inclusive of both 0 and 255.		
Service not message	Indicates the type of the data frame	0	Indicates that the data frame is a Message frame.
		1	Indicates that the data frame is a Service frame.
Request not response	Indicates whether the data frame is a request frame or a response frame.	0	Indicates that the frame is a Response frame.
		1	Indicates that the frame is a Request frame.

Node ID	1: The Node ID consists of 7 bits, where 0 is a reserved ID representing an unknown node. 2: The Node ID values range from 1 to 127, inclusive, with 126 and 127 being reserved IDs. 3: Node ID is divided into Source Node ID and Destination Node ID. 4: Source Node ID represents the ID of the node itself. 5: Destination Node ID represents the ID of the other node. 6: Only Service frames have a Destination Node ID, as they require a response.
---------	--

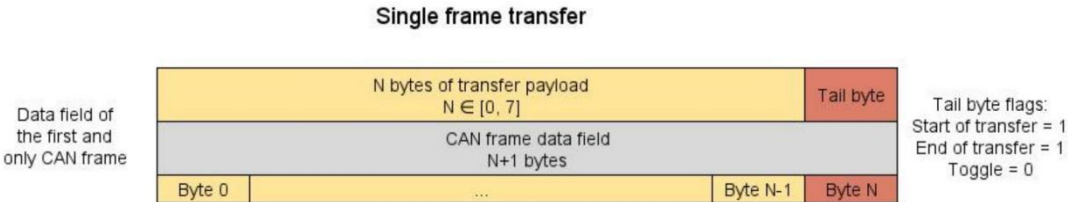
4.1.3 CAN Payload

CANBus 2.0B specifies that each CAN bus frame can transmit up to 8 bytes of data. The TM-UAVCAN protocol divides the 8-byte payload into two parts: Transfer Payload and Tail byte, as shown in the diagram below:



Some data packets may have valid data that exceeds 7 bytes. The TM-UAVCAN protocol specifies that data frames of up to 7 bytes use the Single Frame Transfer format, while data frames exceeding 7 bytes use the Multi-Frame Transfer format. Below are the definitions of Single Frame Transfer and Multi-Frame Transfer.

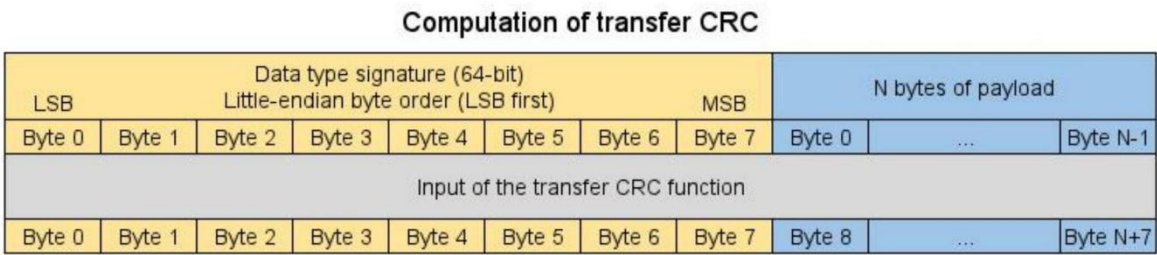
Single frame Transfer: Single Frame Transfer is used to handle data streams where the transmitted data does not exceed 7 bytes.



Multi frame Transfer: Multi-Frame Transfer is used to handle data streams where the transmitted data exceeds 7 bytes. This introduces two concepts:

1. CRC (Cyclic Redundancy Check)
2. Inversion Bit

As shown in the diagram below, CRC is calculated for all payload data and requires the addition of a data signature.



Note: The CRC calculation method with data signature is as shown in the diagram above. Add a 64-bit data signature value before the valid data (N bytes of payload), then use the CRC algorithm provided in the appendix to compute the checksum from Signature Byte 0 to Payload Byte N-1. When calculating this CRC value using the

appendix algorithm, follow these steps:

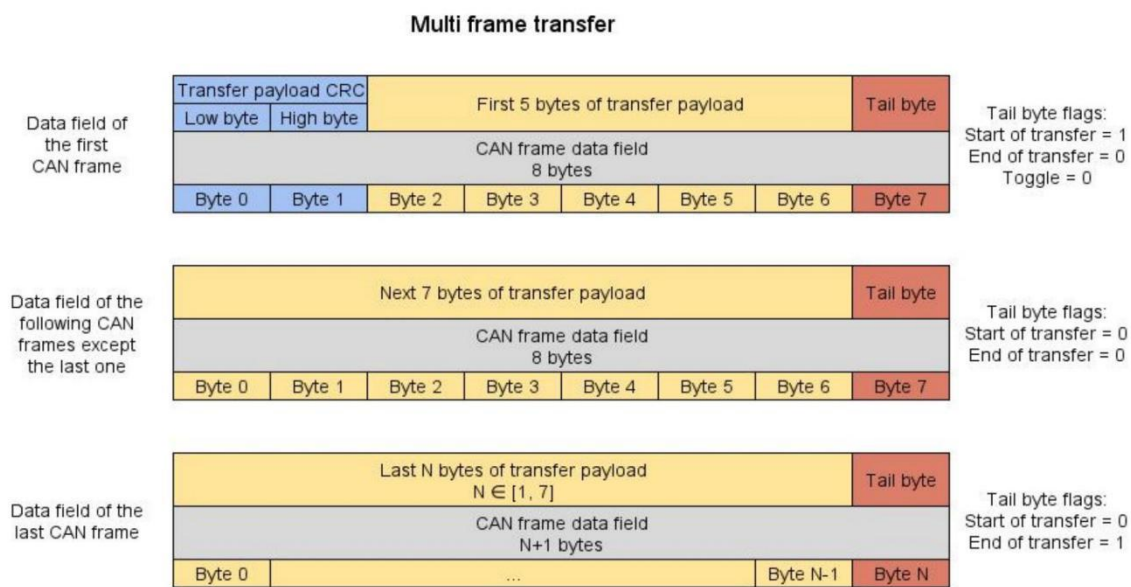
1. Add the 64-bit data signature:

```
`SignatureCrc = crcAddSignature (0xFFFF, SignatureValue);`
```

2. Add the Payload segment data:

```
`CRC = crcAdd(SignatureCrc, NBytesOfPayload, N);`
```

Multi frame Transfer :



Glossary of Terms

Terms	Explanation and Description
Data Frame	1: CAN can transmit up to 8 bytes of data in a single frame.
Data Packet	1: A data packet may contain multiple data frames, with the number of frames determined by the amount of data being transmitted.
Transfer Payload	1: Each data frame in the transmission carries valid data ranging from 0 to 7 bytes.
Tail byte	1: In each data frame, the last byte of the Payload contains additional protocol layer field information.
Signature	1: Each different data frame type has a 64-bit data signature value. 2: When a data packet requires multi-frame transmission, this signature value must be used in the CRC check. 3: Specific values for each data type can be found in the data frame type table in the ESC communication section.
Start of transfer	1: For Single Frame Transfer, the Start of Transfer bit is always set to 1. 2: For Multi-Frame Transfer, if the current frame is the first frame of the data packet, this bit is set to 1; otherwise, it is set to 0.
End of transfer	1: For Single Frame Transfer, the End of Transfer bit is always set to 1. 2: For Multi-Frame Transfer, if the current frame is the last frame of the data packet, this bit is set to 1; otherwise, it is set to 0.
Toggle bit	1: For Single Frame Transfer, the Toggle Bit is always set to 0. 2: For Multi-Frame Transfer, the Toggle Bit is set to 0 in the first frame of the data packet. For

	each subsequent frame, this bit alternates between 0 and 1.
Transfer ID	1: Value range: 0 to 31. 2: For data with the same Data Type ID, each time a data packet is sent, the Transfer ID is incremented by 1, with values ranging from 0 to 31 in a cyclic manner. 3: For multiple data frames within the same data packet, this value remains unchanged.

4.2 CAN ESC Working Mechanism

All ESCs are connected via the CAN bus, with common connection topologies being T-shaped or star-shaped. The throttle input can be provided using PWM for analog throttle or CAN for digital throttle. You can set a priority between PWM throttle and CAN throttle. For example, if PWM throttle is set to take priority, it will be used first. If PWM throttle is abnormal but CAN throttle is functioning normally, the system will automatically switch to CAN throttle. If both PWM and CAN throttles are abnormal, the system will consider the throttle as faulty.

4.2.1 Features

- . The ESC has its own unique serial number (SN).
- . When powered on, the ESC performs an automatic self-check. The self-check status can be queried by sending a self-check query command.
- . Supports setting the ESC node ID.
- . Supports setting the CAN bus baud rate.
- . The data reporting rate is adjustable.
- . Other features depend on the specific ESC model.
- . Analog and digital dual throttle input.

4.2.2 Bus Bandwidth

TM-UAVCAN supports various bus baud rates. You can set the rate according to the number of devices and the cable configuration, with the default rate being 1 MHz.

4.3 Data Frame Type List

4.3.1 Data Frame Type List

Data Frame Type List					
Frame Types	Frame ID (DEC)	Signature Code	Priorit y	Maximum Data Field Length	Data Frame Description
Messeage	1030	0x217F5C87D7EC951D	LOW	36Byte	Raw Throttle Command Data for Channels 1-20
Messeage	1033	0x948F5E0B33E0EDEE	LOW	27Byte	ESC Parameter Setting Command
Messeage	1034	0xA9AF28AEA2FBB254	LOW	14Byte	ESC Operational Status Feedback Data
Messeage	1038	0xCE2B6D6B6BDC0AE8	LOW	260Byte	Commands Issued from the Ground Control Station to the ESC
Messeage	1039	0xAACF9B4B2577BC6E	LOW	260Byte	ESC Data Feedback Command
Messeage	1332	0x462875A0ED874302	LOW	74Byte	ESC Parameter Setting Feedback Command

4.4 Data Frame Type Description

4.4.1 RawCommand(1030)

RawCommand Transmit Frame (N-axis multi-frame, 14*N bit valid data)

Data frame byte index	Parameter name	Number of bytes	Data type	Content
1-N	Throttle command	N(<=20)	INT14	14-bit throttle data, arranged in order from 0 to N, with the remaining bits automatically padded to 8-bit boundaries

After receiving the throttle data, the motor parses the information according to its ESC ID and performs the corresponding action.

Explanation:

- 1: RawCommand sends the throttle command as a broadcast frame, requiring no acknowledgment. All ESCs on the bus receive and parse it simultaneously.
- 2: Each throttle channel occupies 14 bits, with the highest bit (bit 13) as the sign bit. The digital throttle range is -8191 to 8191, where 0 represents no throttle and 8191 represents full throttle. Negative throttle values are not supported by the protocol; negative values are considered throttle anomalies, and the status bit will return a throttle error. If the erroneous throttle does not reach the timeout period, the last valid throttle will be maintained. If the timeout is reached, the throttle will automatically return to zero.
- 3: Users should select the N-axis RawCommand data frame based on the actual number of ESCs. It is recommended to connect a maximum of 8 CAN nodes per CAN channel to ensure communication quality.
- 4: RawCommand Data Transmission Format Analysis (Example for a Quadrotor):
 - a: Assume we need to send a throttle value of 1000 (0x03E8) to four channels.
Data to be sent (HEX): {3E8, 3E8, 3E8, 3E8}
Memory format (HEX): {E8, 03, E8, 03, E8, 03, E8, 03}
Memory format (BIN): {11101000, 00000011, 11101000, 00000011, 11101000, 00000011, 11101000, 00000011}
 - b: Throttle transmission involves converting the original 16-bit throttle data to 14-bit throttle data (removing the highest 2 bits of the original 16-bit data).
Original 16-bit data:
HEX: {0xE8, 0x03, 0xE8, 0x03, 0xE8, 0x03, 0xE8, 0x03}
BIN: {1110_1000_0000_0011_ 1110_1000_0000_0011_ 1110_1000_0000_0011_ 1110_1000_0000_0011_}
Convert to 14-bit data (remove the highest 2 bits of the original data's high byte, as highlighted in red):
HEX: {1110_1000_0000_1111_1010_0000_0011_ 1110_1000_0000_1111_1010_0000_0011_}
BIN: {0xE8, 0x0F, 0xA0, 0x3E, 0x80, 0xFA, 0x03}
- 5: The mapping relationship between CAN digital throttle and PWM throttle is as follows:
CAN throttle period: 2.5 ms, 400 Hz
PWM throttle frequency: 50-500 Hz
Spraying Drone: Throttle pulse width range is 1050-1950 μ s
UAV: Throttle pulse width range: 1040-1940 μ s

a:Throttle data is represented in 14 bits with a range of -1 to 1 (-8191 to +8191) (resolution of 1/16384). Zero represents 0 throttle, 1 represents maximum clockwise throttle, and -1 represents maximum counterclockwise throttle. Currently, the ESC supports both PWM and CAN throttle control methods. For traditional PWM throttle, the fixed throttle pulse width is 1040 μ s to 1940 μ s (the recognizable throttle pulse width range is 800-2200 μ s). For CAN digital throttle, negative throttle is not currently supported (negative digital throttle will result in a throttle error).

B:The ESC internally maps the digital throttle range of 0-8191 to a PWM throttle range of 1040 μ s to 1940 μ s, and then controls the motor's rotation. When set to CAN throttle mode, the original PWM input will produce a PWM pulse width corresponding to the digital throttle.

4.4.2 ParamCfg(1033)

ParamCfg Send frame (27 bytes of valid data)

Parameter setting send command(1033)

No	Parameter	Byte count	Data Type	Content
1	Esc ID	1	UINT8	Esc ID0-19
2-5	ESC to be configured UUID	4	UINT32	Unique Esc 32Bits ID
6-7	Esc ID	2	UINT16	Configure Esc ID
8-9	Over-voltage protection threshold	2	UINT16	Set Over-voltage protection threshold
10-11	Over-current protection threshold	2	UINT16	Set Over-current protection threshold
12-13	Over-temperature protection threshold	2	UINT16	Set Over-temperature protection threshold
14-15	Acceleration limit	2	UINT16	Set Acceleration limit
16-17	Deceleration limit	2	UINT16	Set Deceleration limit
18-19	Rotation direction	2	UINT16	Set Rotation direction
20	Timing setting	1	UINT8	Set the timing, with a range of 1-29 mapping to 1-29°.
21	Throttle priority and Rotor Lock	1	UINT8	Enable Lock: high four bits 1000 Disable Lock: high four bits 0000 PWM: low four bits 0001 CAN: low four bits 0010
22-23	LED Seting	2	UINT16	0-2 bits: RGB light enable switch; 3 bit: light flashing switch; 4-15 bits: light flashing frequency (0.1 Hz)
24	Bus rate	1	UINT8	0-1Mbps 1-500Kbps 2-250Kbps 3-125Kbps 4-100Kbps 5-50Kbps
25-26	Data reporting rate	2	UINT16	Reporting rate 0-400HZ
27	Save	1	UINT8	0-Temporary set 1-Permanent set

Explanation:

1: Set ESC parameters; 0xFF is the default value, and if the current value is 0xFF/0xFFFF, no modification is made. After

setting is complete, respond with the parameter setting feedback command (1332).

2: You can send a data frame with all values set to 0xFF to retrieve parameter information from all ESCs on the bus.

4.4.3 ESC Status(1034)

EscStatus Return Frame (14Byte valid data)

No	Parameter	Byte count	Data Type	Content
1-4	Status Bit	4	UINT32	Status Bit
5-6	Voltage Value	2	FLOAT16	Voltage Value, in volts
7-8	Current Value	2	FLOAT16	Current Value, in amperes
9-10	Temperature Value	2	FLOAT16	Temperature Value, in Kelvin
11-13	Speed Value	2~3	INT18	Speed Value, in RPM
13-14	Throttle Value	<1	UINT7	Throttle Value, 0-100 represents 0-100%
14	ESC Serial Number	<1	UINT5	ESC Number, 0-19 represents channels 1-20

Status Byte Structure:

Status Byte Structure Table							
Bit16-31	Bit12-15			Bit11	Bit10	Bit9	Bit8
Encoder value 0-16383 represents 0-360°	1-Off mode			Encoder error	Low bridge fault	Up bridge fault	Operational amplifier fault
	2-Idle mode						
	3-Soft start mode						
	4-Operating mode						
	5-Coast-down mode						
	6-Error mode						
	7-Low-speed forward rotation propeller folding mode						
	8-Low-speed reverse folding mode						
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Stall	Capacitor Overtemperature	MOS Overtemperature	Throttle Abnormal	Throttle Loss	Overcurrent	Undervoltage	Overvoltage

Explanation:

1: The `EscStatus` sending frame is a broadcast frame and does not require acknowledgment. All nodes on the bus receive it simultaneously, and only the flight controller or programmer participates in the analysis.

2: Each ESC provides status data according to its node ID and ESC number. It is necessary to pre-set each ESC's ID to avoid conflicts.

3: Users should set the return rate based on actual needs to avoid high bus load and processing load caused by excessively high reporting rates.

4.4.4 PUSHSCI(1038)

PUSHSCI Send frame (up to 260 bytes of valid data)

PUSHSCI Host computer transmits data to ESC command data structure

No	Parameter	Byte count	Data Type	Content
1-4	Data sequence number	4	UINT32	Data packet sequence number
5-N	Raw data value	0-255	UINT8	Data packet data

Data packet structure:

1. Set the motor's current position to zero

No	Parameter	Byte count	Data Type	Content
1-2	Frame header	2	UINT16	0xEC、0x96
3	Frame ID	1	UINT8	0x08
4	Frame count	1	UINT8	0-255 cyclic increment
5	Power unit ID	1	UINT8	0xA1~0xA9: corresponding to 1~9
6	Frame Length	1	UINT8	0x07
7	Checksum	1	UINT8	The sum of bytes (1~6) takes the lower 8 bits.

After receiving the command, the motor sets the current position as the zero point.

2.FOC parameter setting data packet

No	Parameter	Byte count	Data Type	Content
1-2	Frame header	2	UINT16	0xEC、0x96
3	Frame ID	1	UINT8	0x1E
4	Frame count	1	UINT8	0-255 cyclic increment
5	Power unit ID	1	UINT8	The aircraft has 9 power units. 0xA1~0xA9 correspond to units 1~9. The ESC addresses in the power system can be set to correspond to the actual assembly positions. (28027 defaults to 0xA1; 103 should be set according to the actual ID.)
6	Frame Length	1	UINT8	0x1F
7-8	ESC ID	2	UINT16	Set ESC ID
9-10	Over-voltage protection threshold	2	UINT16	Set Over-voltage protection threshold
11-12	Over-current protection threshold	2	UINT16	Set Over-current protection threshold
13-14	Over-temperature protection threshold	2	UINT16	Set Over-temperature protection threshold
15-16	Beep volume	2	UINT16	Set Beep volume
17-18	Acceleration limit	2	UINT16	Set Acceleration limit
19-20	Deceleration limit	2	UINT16	Set Deceleration limit
21-22	Rotation direction	2	UINT16	Set Rotation direction
23-24	Throttle priority	2	UINT16	1: PWM 2: CAN

				(24-bit: Position Loop Enabled)
25-26	LED Settings	2	UINT16	0-2 bits: RGB light enable switch; 3 bit: light flashing switch; 4-15 bits: light flashing frequency (0.1 Hz)
27	Bus rate	1	UINT8	0-1Mbps 1-500Kbps 2-250Kbps 3-125Kbps 4-100Kbps 5-50Kbps
28-29	Data report rate	2	UINT16	Feedback rate: 0-400 Hz
30	Save options	1	UINT8	0-Temporary settings 1-Permanent settings
31	Checksum	1	UINT8	Checksum: Sum of bytes (1 to 28), taking the lower 8 bits

3. FOC parameter and status information retrieval data packet

No	Parameter	Byte count	Data Type	Content
1-2	Frame header	2	UINT16	0xEC、0x96
3	Frame ID	1	UINT8	0x1A
4	Frame count	1	UINT8	0-255 cyclic accumulation
5	Power unit ID	1	UINT8	0xA1~0xA9: Correspond to 1~9, or 0xFF to retrieve all parameters on the bus.
6	Frame length	1	UINT8	7
7	Checksum	1	UINT8	The checksum is the lower 8 bits of the sum of bytes 1 to 6.

4. Controller output control data frame (Controller → Power System).

No	Parameter	Byte count	Data Type	Content
1-2	Frame header	2	UINT16	0xEC、0x96
3	Frame ID	1	UINT8	0x06
4	Frame count	1	UINT8	0-255 cyclic accumulation
5	Power unit ID	1	UINT8	There are 9 power units on the aircraft. 0xA1 to 0xA9: Correspond to motor units 1 to 9. The ESC addresses in the power system can be set to correspond to the actual assembly positions.
6	Frame length	1	UINT8	0x0C
7	Control mode	1	UINT8	0x00: No rotor locking mode, responds normally to flight control PWM and also serves as the power system query command. 0xEE: Motor low-speed forward rotation and blade retraction. 0x22: Motor low-speed reverse rotation and blade retraction. 0x88: Rotor locks current position. 0x66: Motor is in free mode. 0x55: Motor duty cycle loop. 0x5A: Motor reverse duty cycle loop

				0x44: Motor current loop 0x4A: Motor reverse current loop 0x33: Motor speed loop 0x3A: Motor reverse speed loop 0x11: Motor position loop 0x1A: Motor reverse position loop 0xBB: Motor current brake loop
8-9	Motor Control Type Throttle duty/ speed/position	2	UINT16	0-100 corresponds to 0-100% throttle 0-10000 corresponds to 0-10000 RPM 0-16383 corresponds to 0-360°
10-11	Reserved	2	UINT16	0x00
12	Checksum	1	UINT8	Checksum: Sum of bytes 1 to 11, taking the lower 8 bits

Explanation:

1. The PUSHSCI transmission data to ESC command frame is a broadcast frame and does not require acknowledgment. All nodes on the bus receive and parse it simultaneously.
2. The data frame contains an ID field. Each ESC parses the command according to its node ID and ESC number, so it is necessary to set the ID numbers for each ESC in advance to avoid incorrect responses.
3. Users should set the transmission rate based on actual needs. The total frame rate on the bus should be controlled within 2400 frames (CAN rate 1 Mbps) to avoid high bus and processing load caused by excessive reporting rates.

4.4.5 PUSHCAN(1039)

PUSHCAN return frame (up to 260 bytes of valid data)

PUSHCAN ESC data feedback instruction data structure

No	Parameter	Byte count	Data Type	Content
1-4	Data serial number	4	UINT32	Data packet sequence number
5-N	Raw data value	0-255	UINT8	Data packet data

Data packet structure:

1. Power system feedback status data frame (Power system → Controller)

No	Parameter	Byte count	Data Type	Content
1-2	Frame header	2	UINT16	0x7B、0x8C
3	Frame ID	1	UINT8	0x15
4	Frame count	1	UINT8	0-255 cyclic accumulation
5	Power unit ID	1	UINT8	There are 9 power units on the aircraft. 0xA1~0xA9 correspond to units 1~9. The ESC addresses in the power system can be set to match the actual assembly positions.
6	Frame length	1	UINT8	0x1C
7	System status of the power unit ID	1	UINT8	0x00: No rotor lock state 0x11: Rotor lock state 0xCC: System fault
8-9	Motor current position	2	UINT16	0-360°, resolution: 0.01°
10-11	Receive PWM value	2	UINT16	resolution: 0.1us
12	Actual output throttle	1	UINT8	resolution: 0.4%

	value			
13-14	Motor speed	2	INT16	resolution: 1rpm
15-16	Voltage	2	UINT16	resolution: 0.01V
17-18	Current	2	INT16	resolution: 0.01A
19-20	Temperature	2	UINT16	resolution: 0.01℃
21-22	Motor fault	2	UINT16	Error flag data
23	Motor status	1	UINT8	0x00
24-25	ESC power-on count	2	UINT16	Unit:times.
26-27	The runtime of the ESC.	2	UINT16	Unit: seconds
28	Checksum	1	UINT8	The sum of bytes 1 to 27, taking the lower 8 bits

2. FOC parameter setting feedback data packet

No	Parameter	Byte count	Data Type	Content
1-2	Frame header	2	UINT16	0x7B、0x8C
3	Frame ID	1	UINT8	0x1E
4	Frame count	1	UINT8	0-255 cyclic accumulation
5	Power unit ID	1	UINT8	There are 9 power units on the aircraft. 0xA1~0xA9 correspond to units 1~9. The ESC addresses in the power system can be set to match the actual assembly positions.
6	Frame length	1	UINT8	0x37
7-22	ESC hardware version	16	UINT16	ESC version number
23-38	ESC software version	16	UINT16	ESC software version
39-40	Acceleration limit	2	UINT16	Acceleration limit
41-42	Deceleration limit	2	UINT16	Deceleration limit
43-44	Rotation direction	2	UINT16	Rotation direction
45-46	Position loop enable flag	2	UINT16	Position loop enable flag
47-48	Throttle priority	2	UINT16	1: PWM 2: CAN
49-50	LED setting	2	UINT16	0-2 bits: RGB light enable switch, 3 bit: light blinking switch, 4-15 bits: light blinking frequency (0.1 Hz)
51	Bus rate	1	UINT8	0-1Mbps 1-500Kbps 2-250Kbps 3-125Kbps 4-100Kbps 5-50Kbps
52-53	Data feedback rate	2	UINT16	Return rate: 0-400Hz
54	Save options	1	UINT8	0-Temporary settings 1-Permanent setting 2-
55	Checksum	1	UINT8	(1~54) Take the lower 8 bits of the sum of bytes

4.4.6 ParamGet(1332)

ParamGet returns frame (maximum 74Byte valid data)

Parameter setting report command (1332) data structure

No	Parameter	Byte count	Data Type	Content
1	ESC ID	1	UINT8	ESC ID0-19
2-5	UUID of the ESC to be configured	4	UINT32	Unique 32-bit ID of the ESC
6-7	ESC ID	2	UINT16	To obtain the ESC ID, 0XFF means simultaneous feedback from all ESCs on the bus.
8-9	Over-voltage protection threshold	2	UINT16	Get the ESC over-voltage protection threshold
10-11	Over-current protection threshold	2	UINT16	Get Over-current protection threshold
12-13	Over-temperature protection threshold	2	UINT16	Get Over-temperature protection threshold
14-15	acceleration limit	2	UINT16	Get acceleration limit
16-17	Deceleration limit	2	UINT16	Get deceleration limit
18-19	Rotation direction	2	UINT16	Get rotation direction
20	Timing setting	1	UINT8	Get the timing, with 1-29 mapping to 1-29°.
21-22	Power-up count	2	UINT16	Get the total number of times the ESC is powered on
23-26	Power on time	4	UINT32	Get the total power-on time of the ESC (seconds)
27-30	Production time	4	UINT32	Get the ESC production date
31-34	Number of failures	4	UINT32	Get the cumulative number of ESC failures
35	Throttle priority and Rotor Lock	1	UINT8	Enable Lock: high four bits 1000 Disable Lock: high four bits 0000 PWM: low four bits 0001 CAN: low four bits 0010
36-37	LED setting	2	UINT16	0-2bit: RGB light enable switch, 3bit: light flashing switch, 4-15bit: light flashing frequency (0.1HZ)
38	Bus rate	1	UINT8	0-1Mbps 1-500Kbps 2-250Kbps 3-125Kbps 4-100Kbps 5-50Kbps
39-40	Data report rate	2	UINT16	Report rate 0-400HZ
41	Save options	1	UINT8	0-Temporary settings 1-Permanent settings
42-73	Reserved data	32	UINT32	Manufacturer reserved data

Explanation:

1. The ParamGet parameter retrieval return instruction is a broadcast frame and does not require a response. All nodes on the bus receive it simultaneously.
2. The data frame includes an ID bit, and only the flight controller or parameter adjuster needs to parse and process the data. This frame is for parameter setting and retrieval feedback. Users can select as needed.
3. Users should set the transmission rate according to actual needs. During operation, the total frame rate on the bus should be controlled within 2400 frames (CAN rate 1Mbps). When not operating, the total frame rate should be controlled within 4000 frames (CAN rate 1Mbps) to avoid high bus load and processing load due to excessive feedback rates.

5 Attachment

5.1. CRC support:

CRC algorithm description:

Name: CRC- 16-CCITT-FALSE

Description: <http://reveng.sourceforge.net/crc-catalogue/16.htm#crc.cat.crc-16-ccitt-false>

Initial value: 0xFFFF

Poly: 0x1021

Reverse: no

Output XOR: 0

Check: 0x29B1

```
/*
 * CRC functions
 */
uint16_t crcAddByte(uint16_t crc_val, uint8_t byte)
{
    crc_val ^= (uint16_t) ((uint16_t) (byte) << 8);
    for (uint8_t j = 0; j < 8; j++)
    {
        if (crc_val & 0x8000U)
        {
            crc_val = (uint16_t) ((uint16_t) (crc_val << 1) ^ 0x1021U);
        }
        else
        {
            crc_val = (uint16_t) (crc_val << 1);
        }
    }
    return crc_val;
}

uint16_t crcAddSignature(uint16_t crc_val, uint64_t data_type_signature)
{
    for (uint16_t shift_val = 0; shift_val < 64; shift_val = (uint16_t)(shift_val + 8U))
    {
        crc_val = crcAddByte(crc_val, (uint8_t) (data_type_signature >> shift_val));
    }
    return crc_val;
}
```

```

uint16_t crcAdd(uint16_t crc_val, const uint8_t* bytes, size_t len)
{
    while (len--)
    {
        crc_val = crcAddByte(crc_val, *bytes++);
    }
    return crc_val;
}

```

5.2. float16 support:

```

/*
UAVCAN floating-point data type conversion
*/
union FP32
{
    uint32_t u;
    float f;
};

const union FP32 f32inf = { 255UL << 23 };
const union FP32 f16inf = { 31UL << 23 };
const union FP32 magic   = { 15UL << 23 };
const uint32_t sign_mask = 0x80000000U;
const uint32_t round_mask = ~0xFFFFU;

uint16_t ConvertFloatToFloat16 ( float value)
{
    union FP32 in;
    in.f = value;
    uint32_t sign = in.u & sign_mask;
    in.u ^= sign;
    uint16_t out = 0;

    if (in.u >= f32inf.u)
    {
        out = (in.u > f32inf.u) ? (uint16_t)0x7FFFU : (uint16_t)0x7C00U;
    }
}

```

```

else
{
    in.u &= round_mask;
    in.f *= magic.f;
    in.u -= round_mask;
    if(in.u > f16inf.u)
    {
        in.u = f16inf.u;
    }
    out = (uint16_t)(in.u >> 13U);
}
out |= (uint16_t)(sign >> 16U);
return out;
}

float ConvertFloat16ToFloat(uint16_t value)
{
    const union FP32 magic = { (254UL - 15UL) << 23U };
    const union FP32 was_inf_nan = { (127UL + 16UL) << 23U };
    union FP32 out;

    out.u = (value & 0x7FFFU) << 13U;
    out.f *= magic.f;
    if (out.f >= was_inf_nan.f)
    {
        out.u |= 255UL << 23U;
    }
    out.u |= (value & 0x8000UL) << 16U;
    return out.f;
}

```

5.3 DSDL configuration reference

5.3.1 1030.RawCommand

```
struct { uint8_t len; int14_t data[20]; }cmd;
```

5.3.2 1033.ParamCfg

```
uint8 esc_index  
uint32 esc_uuid  
uint16 esc_id_set  
uint16 esc_ov_threshold  
uint16 esc_oc_threshold  
uint16 esc_ot_threshold  
uint16 esc_acc_threshold  
uint16 esc_dacc_threshold  
int16 esc_rotate_dir  
uint8 esc_timing  
uint8 esc_signal_priority  
uint16 esc_led_mode  
uint8 esc_can_rate  
uint16 esc_fdb_rate  
uint8 esc_save_option
```

5.3.3 1034.ESC_STATUS

```
uint32 error_count  
float16 voltage  
float16 current  
float16 temperature  
int18 rpm  
uint7 power_rating_pct  
uint5 esc_index
```

5.3.4 1038.PUSHSCI

```
uint32 data_sequence  
struct { uint8_t len; uint8_t data[255]; }data;
```

5.3.5 1039.PUSHCAN

```
uint32 data_sequence  
struct { uint8_t len; uint8_t data[255]; }data;
```

5.3.6 1332.ParamGet

```
uint8 esc_index
uint32 esc_uuid
uint16 esc_id_req
uint16 esc_ov_threshold
uint16 esc_oc_threshold
uint16 esc_ot_threshold
uint16 esc_acc_threshold
uint16 esc_dacc_threshold
int16 esc_rotate_dir
uint8 esc_timing
uint16 esc_startup_times
uint32 esc_startup_duration
uint32 esc_product_date
uint32 esc_error_count
uint8 esc_signal_priority
uint16 esc_led_mode
uint8 esc_can_rate
uint16 esc_fdb_rate
uint8 esc_save_option
struct { uint8_t len; uint8_t data[32]; }rsvd;
```

5.4 Reference link

- 1.Dronecan project:<https://dronecan.github.io>
- 2.Dronecan configuration example:<https://github.com/dronecan/libcanard/tree/master/examples>
- 3.ardupilot project:<https://ardupilot.org>
- 4.PX4 project:<https://px4.io>

6 Version history

Date	Version	Change log
2024.5.16	V2.1	The first version of the protocol is released
2024.7.3	V2.2	1.Fixed some document layout errors 2.Added version history 3.Deleted some unavailable data frame types in Chapter 4.3
2025.03.20	V2.3	1、 Changed the temperature in ESC Status from Celsius to Kelvin. 2、 Changed the description of the data frame type in Section 4.3.1. 3、 Fixed the motor control type in the PUSHSCI(1038) data. 4、 Fixed 21-byte throttle priority in ParamCfg(1033) data