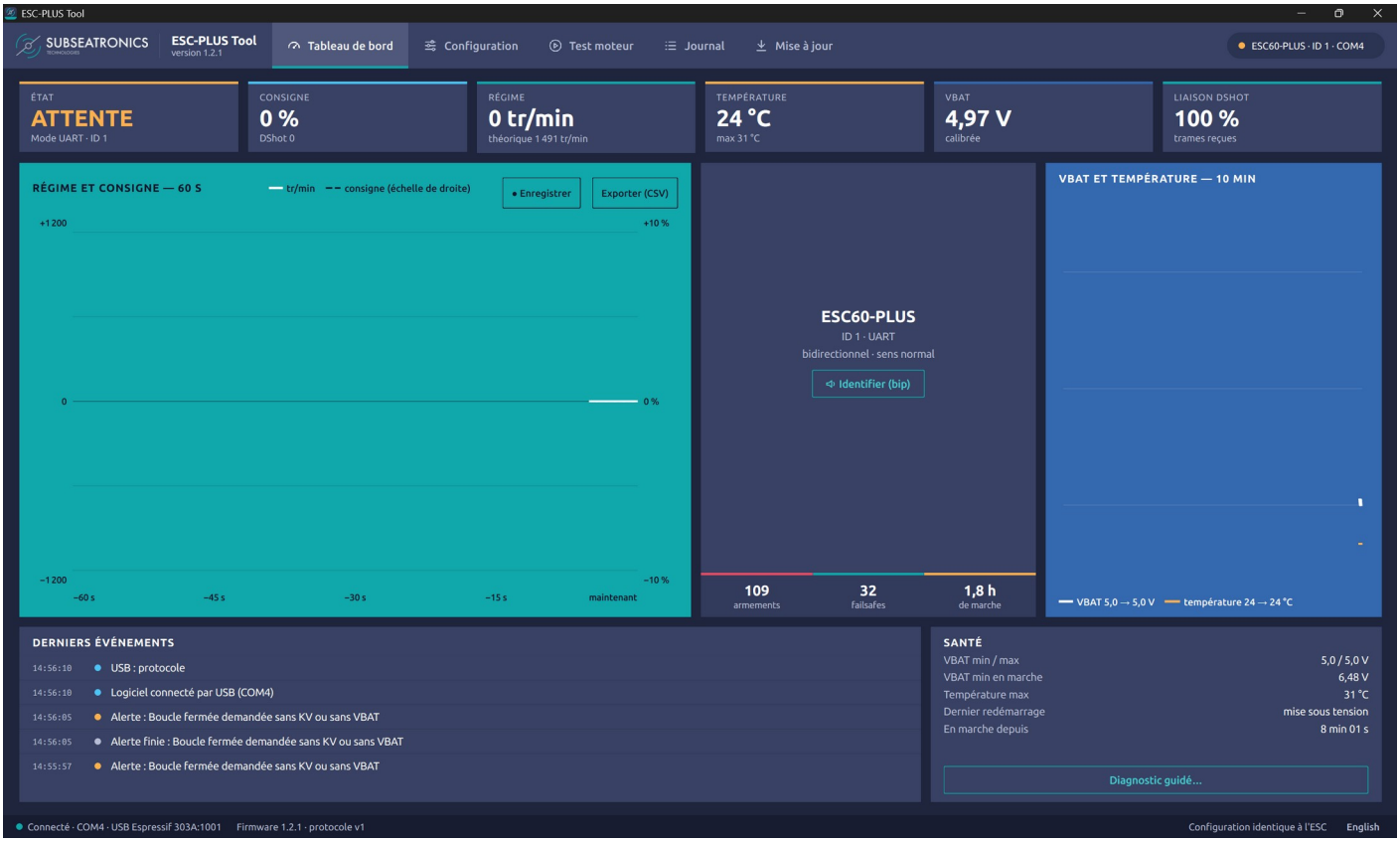
ESC60-PLUS

Integration Manual

Controlling the ESC from your controller: UART and CAN protocols

Document version 1.0 - Subseatronics Technologies

Firmware ESC-PLUS 1.2.1 - October 19, 2026



SUBSEATRONICS
TECHNOLOGIES

Contents

1. Overview 3
2. Wiring 4
2.1 Control connector (connector B) 4
2.2 Serial link (UART) 4
2.3 CAN bus 4
2.4 Power supply and power-up 5
3. Principles common to UART and CAN 5
3.1 ESC states 5
3.2 Arm, command, stop 5
3.3 Setpoint loss: T1, T2 and recovery 6
3.4 Throttle or speed 6
3.5 Faults (the motor stops) and alerts (it keeps running) 7
3.6 Commands must be resent if they get no response 7
4. UART protocol 8
4.1 Format 8
4.2 Commands 8
4.3 TEL: telemetry 8
4.4 EVT: events and other lines sent by the ESC 9
4.5 Several ESCs on one link: @ID addressing and RS-485 bus 9
4.6 Typical sequence (direct link) 10
5. CAN protocol 11
5.1 Identifiers 11
5.2 Command frame 0x100 + ID (controller → ESC) 11
5.3 Telemetry frame 0x180 + ID (ESC → controller, 8 bytes) 11
5.4 Status frame 0x1C0 + ID (ESC → controller, 8 bytes) 12
5.5 Grouped frames: four ESCs per frame (option) 13
5.6 Special bus behaviors 13
5.7 Typical sequence (ID 3) 13
6. PWM and analog modes (reminder) 14
7. Signaling: LEDs and beeps 14
8. What to know about the motor and the telemetry 15
9. ESC configuration (ESC-PLUS Tool, over USB) 15
10. Quick reference 16

1. Overview

The ESC60-PLUS is a speed controller for ROV brushless thrusters. It contains two processors: a power stage driven by the Bluejay firmware (motor commutation) and a supervision microcontroller (ESP32-C3) that receives your command, applies the safety functions and sends the telemetry back to you. This supervisor is what you control.

Four command modes, to choose from, set in ESC-PLUS Tool:

Mode	Signal	Intended for
RC PWM	1100 / 1500 / 1900 µs pulses (radio receiver, servo controller)	Simple control without feedback
Analog	0 to 5 V (0.5 V reverse, 2.5 V neutral, 4.5 V forward)	Joystick or analog output of a PLC
UART	3.3 V text serial link, 115,200 baud (57,600 or 230,400)	Controller with a serial port; several ESCs possible on an RS-485 bus (chapter 5)
CAN	Classic CAN bus 500 kbit/s (250 or 1000), 11-bit identifiers	Controller with a CAN port; up to 31 ESCs on the same bus

With UART and CAN, your controller can request a **throttle** (-100% to +100%, open loop) or a **speed** (rpm, closed loop regulated by the ESC), arm and disarm the ESC, read the telemetry (state, speed, temperature, voltage, faults) and trigger identification beeps.

Five rules to remember

1. **Arm before commanding**: the ESC only follows a setpoint in the Armed state (ARM command, then 2 s of arming).
2. **Keep the setpoint alive**: resend it at least **10 times per second**. Without a valid setpoint for 500 ms, the motor is forced to 0; after 5 s, the ESC disarms (failsafe) and must be re-armed.
3. **Resend a command that gets no response**: if the ESC does not answer within 200 ms (UART) or if the status frame does not reflect the command (CAN), send it again. See 3.6.
4. **Read the fault and alert codes**: they are in every `TEL :` line (UART) and in the status frame (CAN); a fault stops the motor, an alert does not.
5. **Ignore what you do not recognize**: future versions add fields at the end of lines and new messages, they never remove any.

2. Wiring

The ESC has two identical JST-SH connectors (1.0 mm pitch, 8 pins): **CN1** on the TOP side for control (this is connector B, detailed in 2.1) and **CN2** on the BOTTOM side for the configuration USB and the programming of MCU 2 (power stage, Bluejay). Pinout and wire colors:

Pin	CN1 (TOP side): control	Wire	CN2 (BOTTOM side): USB and programming	Wire
1	UART_TX (ESC serial output)	Purple	C2D (MCU 2 programming)	Purple
2	UART_RX (ESC serial input)	Orange	C2CK (MCU 2 programming)	Orange
3	GND	White	VBUS (+5 V from USB)	USB plug
4	CAN_H	Blue	GND	USB plug
5	CAN_L	Green	NC (do not connect)	-
6	GND	Yellow	USB D-	USB plug
7	PWM_IN (RC PWM input)	Black	USB D+	USB plug
8	0/10V_IN (analog input)	Red	GND	USB plug

Caution: this color code does not follow the RC servo convention. On CN1, the PWM signal is the black wire (pin 7) and its ground is the yellow wire (pin 6); the red wire (pin 8) is the analog input, not a power supply. On CN2, the USB plug is already fitted at the end of the cable.

2.1 Control connector (connector B)

JST-SH connector, 1.0 mm pitch, 8 pins. Both mounting pins are grounded.

Pin	Name	Use
1	UART_TX	ESC serial output → RX input of your controller (3.3 V)
2	UART_RX	ESC serial input ← TX output of your controller (3.3 V only, never 5 V)
3	GND	Ground
4	CAN_H	CAN bus
5	CAN_L	CAN bus
6	GND	Ground
7	PWM_IN	RC PWM input
8	0/10V_IN	Analog input (0-5 V on the current board)

2.2 Serial link (UART)

- Logic levels **3.3 V**, 8 data bits, no parity, 1 stop bit (8N1). Baud rate set in ESC-PLUS Tool: 57,600, **115,200** (default) or 230,400 baud.
- Point to point: cross TX and RX (ESC pin 1 to the controller RX, pin 2 to its TX), common ground.
- Several ESCs on one link: RS-485 bus with, for each ESC, an RS-485 transceiver **with automatic direction switching** (the ESC does not drive a direction pin), terminations at both ends, and bus mode enabled in each ESC (chapter 5). Never connect the 3.3 V TX outputs of several ESCs directly together.

2.3 CAN bus

- Classic CAN (not CAN FD), 11-bit identifiers, 250, **500** (default) or 1000 kbit/s, the same bit rate on the whole bus.
- One **120 Ω** termination at each end of the bus, and only there. The ESC can carry one (resistor R15 on the board): check on each board whether it is fitted, and keep it only on the ESCs at the ends of the bus. Between CAN_H and CAN_L, with the bus powered off, you should measure about 60 Ω.

- **The CAN transceiver is powered by the power supply (VBAT)**: on the USB cable alone, the ESC answers the software but the CAN bus is inactive (alert `CAN_BUSOFF`). Run your CAN tests with the power supply connected.

2.4 Power supply and power-up

- The power supply (VBAT) feeds the motor, the supervisor and the CAN transceiver. On USB alone, the supervisor works (configuration, protocol tests) but the motor does not turn and no beep is emitted.
- Set your command **to neutral (setpoint 0) before** connecting the power supply.
- After power-up, the ESC takes about **5 seconds** to be ready (power stage initialization, startup beeps). On UART, it announces this with the line `BOOT:ESC_PLUS_V1.2.1` (installed firmware version); on CAN, the telemetry starts and shows state 1 (waiting).

3. Principles common to UART and CAN

3.1 ESC states

The same state code is used everywhere (UART and CAN telemetry, LEDs, beeps).

Code	State	Motor	How it is entered / left
0	BOOT - startup	Stopped	Power-up; lasts about 5 s
1	WAIT - waiting	Stopped	Idle state. Exit: ARM command
2	ARMING - arming	Stopped	After ARM; lasts the arming time (2 s by default), a high beep at the end
3	ARMED - armed	Follows the setpoint	End of arming. Exit: DISARM, setpoint loss, fault
4	FAILSAFE	Stopped	No setpoint for T2 (5 s). Must be re-armed (ARM)
5	SUSPEND - suspended	Forced to 0	No setpoint for T1 (500 ms); automatic return to Armed (with a 0.5 s ramp) if the setpoint returns before T2
6	FAULT - fault	Stopped	Active fault (table 3.5). Back to waiting when the cause clears; re-arm afterwards
7	TEST - motor test	Driven by the PC	From ESC-PLUS Tool only; ARM is refused (BUSY), and so is a setpoint (NOTARM)

3.2 Arm, command, stop

1. Send **ARM** (UART: `A`; CAN: byte `0x02`). The ESC answers `OK:ARMING` and goes to state 2.
2. Wait for the end of arming: state 3 in the telemetry, line `EVT:ARMED` on UART. During arming, a setpoint is refused (`NOTARM`).
3. Send the setpoint **continuously**, at least 10 times per second, even if it does not change: it is what keeps the motor alive (see 3.3).
4. To stop the motor without disarming: setpoint 0. To stop everything: **DISARM** (UART: `D`; CAN: `0x03`) - immediate effect, even with the motor running.

In bidirectional mode, a positive setpoint turns the motor forward, a negative one in reverse. A change of direction goes through a freewheeling stop (up to 2 s, adjustable) before restarting in the other direction: this is normal, and it is what prevents voltage surges on your power supply.

3.3 Setpoint loss: T1, T2 and recovery

Timeout	Default value (adjustable)	Effect
T1	500 ms (100 to 2000 ms)	No valid setpoint received for T1: motor forced to 0, Suspended state (5). If the setpoint returns before T2, the motor restarts with a 0.5 s ramp (Armed state, `EVT:RESUME`).
T2	5 s (1 to 10 s)	No setpoint for T2: Failsafe state (4), the ESC is disarmed. ARM must be sent again to restart.

Only a **valid setpoint** (throttle or speed, within limits, ESC armed) restarts these timeouts. A query (`Q`, QUERY), a malformed frame or a frame addressed to another ESC does not restart them.

3.4 Throttle or speed

	Throttle setpoint (open loop)	Speed setpoint (closed loop)
UART	`T:<value>` from -1000 to +1000 (‰)	`S:<rpm>` signed, e.g. `S:3000`, `S:-2500`
CAN	`0x01` SET_THROTTLE, int16 little-endian	`0x06` SET_SPEED, int32 little-endian in rpm
Condition	ESC armed	ESC armed + “Speed command (UART, CAN)” option enabled in ESC-PLUS Tool + motor KV entered + VBAT voltage measured
Behavior	The throttle is applied as is (after the configured curve, limits and ramps, neutral by default)	The ESC regulates the measured speed; the target rises with a ramp (2000 rpm/s by default) and falls without limit. Above VMAX (reachable speed = KV × VBAT × 85%, recalculated continuously), the setpoint is applied at VMAX and a `VMAX` error is reported

The last type of setpoint received sets the mode: a throttle setpoint after a speed setpoint switches back to throttle. In unidirectional mode, a negative setpoint is refused on UART (`ERR:RANGE`) and brought back to 0 on CAN (error 5).

3.5 Faults (the motor stops) and alerts (it keeps running)

Faults - code in the 9th field of `TEL:` (UART) and in byte 0 of the status frame (CAN):

Code	Fault	Cause and exit
0	None	
2	Low VBAT	Voltage below the set threshold for 2 s (threshold disabled by default). Exit: voltage back 0.5 V above the threshold
3	Temperature	Power stage temperature above the threshold (80 °C by default) for 2 s. Exit: 10 °C below
4	Blocked propeller	Setpoint above 15% and motor at 0 rpm for 1 s, three attempts. Exit: acknowledgement by ARM or DISARM
5	CAN identifier conflict	Another device sends the telemetry `0x180 + ID` of this ESC. Exit: 10 s after the last foreign frame
6	High VBAT	Voltage above the set threshold for 2 s (disabled by default)

In fault, the ESC is in state 6; an ARM received during a fault is refused (`ERR:FAULT:<code>`). When the cause clears, the ESC goes back to waiting (1): re-arm.

Alerts - bits added together in the 10th field of `TEL:` (UART, in decimal) and in byte 2 of the status frame (CAN):

Bit	Value	Alert	Meaning
0	1	TELEM	Internal motor telemetry lost while running: the displayed speed is no longer reliable
1	2	VBAT_SAT	Voltage outside the measurable range
2	4	CFG_RESET	Configuration unreadable at startup: default values loaded (to be reconfigured)
3	8	CL_FALLBACK	Speed regulation without speed measurement: feedforward only
4	16	CL_NO_KV	Speed setpoint allowed but VMAX unknown (KV at 0 or VBAT missing)
5	32	CAN_BUSOFF	CAN bus in error, automatic recovery in progress
6	64	NO_TEMP	No confirmed temperature for 7 s
7	128	DERATE	Power reduced before a cut-off (temperature within 10 °C of the cut-off, or voltage within 1 V of the low cut-off): ceiling from 100% down to 25%

3.6 Commands must be resent if they get no response

Robustness rule

With the motor stopped and the ESC disarmed, the ESC occasionally saves its counters and its black box to flash memory (starting 2 s after the stop). During such an access (usually less than one millisecond, exceptionally a few tens of milliseconds when the memory reorganizes itself), reception is suspended: a command **being received** at that precise moment can be lost, **on UART with no response at all** (neither `OK:` nor `ERR:`); on CAN, the frame can be ignored. A command received before or after the access is always processed (it may answer a few tens of milliseconds late).

Rule for your controller: **any action command (ARM, DISARM, setpoint) with no response within 200 ms is resent**; on CAN, check in the telemetry that the state has changed (ARM → state 2 within 100 ms) and resend otherwise. A setpoint kept alive 10 times per second corrects itself at the next transmission.

This rule remains good practice in all circumstances (interference, shared RS-485 bus): write your controller so that it never assumes a command has been executed without having seen its response or its effect in the telemetry.

4. UART protocol

4.1 Format

- ASCII text, one command per line, terminated by CR, LF or CR LF. The ESC terminates its lines with CR LF.
- Upper or lower case accepted. At most 32 useful characters per line.
- Responses: `OK:…` (command accepted), `ERR:<code>…` (refused), `TEL:…` (telemetry), `EVT:…` (event), `DIAG:…`, `LOG:…`, `BOOT:…`. Your controller must **ignore any line it does not recognize**.

4.2 Commands

Command	Effect	Possible responses
`T:<v>`	Throttle setpoint, v from -1000 to +1000 (‰). Applied only if the ESC is armed	`OK:T:<v>`; `ERR:NOTARM`; `ERR:RANGE`; `ERR:BADFMT:<line>`
`S:<rpm>`	Speed setpoint, signed (see 3.4)	`OK:S:<v>`; `ERR:VMAX:<vmax>` (applied, but limited to VMAX); `ERR:CLOFF` (option not enabled); `ERR:NOKV` (KV or VBAT missing); `ERR:NOTARM`; `ERR:RANGE`; `ERR:BADFMT:<line>`
`A` or `ARM`	Arm	`OK:ARMING` then, 2 s later, `EVT:ARMED`; `OK:ARMED` if already armed; `ERR:FAULT:<code>`; `ERR:BUSY` (startup or test in progress)
`D` or `DISARM`	Disarm (immediate stop)	`OK:DISARMED`
`Q` or `QUERY`	Immediate telemetry	One line `TEL:`
`I` or `IDENT`	Identification beeps (ESC armed or not, zero setpoint, motor stopped)	`OK:IDENT`; `ERR:RUNNING`
`F` or `DIAG`	Full diagnostics	One line `DIAG:`
`L` or `LOG`	Log of the last 48 events	Lines `LOG:<ms>,<code>,<detail>` then `LOG:END,<count>`

4.3 TEL: telemetry

Sent automatically at **10 Hz** by default (5 or 20 Hz, or off, adjustable), and on `Q`.

```
TEL:ARMED,626,6000,30,100,2400,9599,6000,0,0
```

Field	Content	Example
1	State (name: BOOT, WAIT, ARMING, ARMED, FAILSAFE, SUSPEND, FAULT, TEST)	ARMED
2	Setpoint in ‰ (0 outside the armed state). In throttle mode: the setpoint received; in speed mode: the throttle computed by the regulation	626
3	Measured speed in rpm	6000
4	Power stage temperature in °C	30
5	Internal telemetry quality (0 to 100%)	100
6	VBAT voltage in centivolts (2400 = 24.00 V)	2400
7	VMAX in rpm (0 = unknown)	9599
8	Target speed in rpm, signed (0 with a throttle setpoint)	6000
9	Active fault code (table 3.5; 0 = none)	0
10	Alerts: sum of the bits (table 3.5; 0 = none)	0

A controller that reads only the first fields remains compatible: fields are added at the end of the line from version to version.

4.4 EVT: events and other lines sent by the ESC

Line	When
`BOOT:ESC_PLUS_V1.2.1`	First line after power-up, when the ESC is ready (about 5 s); the version is that of the installed firmware
`EVT:ARMED`	End of arming: setpoints are accepted
`EVT:SUSPEND`	No setpoint for T1: motor forced to 0
`EVT:RESUME`	Setpoint back before T2: recovery with ramp
`EVT:FAILSAFE`	No setpoint for T2: disarmed, re-arm
`EVT:FAULT:<code>` / `EVT:FAULT_END:<code>`	Start / end of a fault (code: VBAT_LOW, TEMP, STALL, ID_CONFLICT, VBAT_HIGH)
`EVT:ALERT:<name>` / `EVT:ALERT_END:<name>`	Start / end of an alert (names from table 3.5)
`DIAG:ST=…,FLT=…,LAST=…,NFLT=…,ALR=…,ERR=…,MODE=…,VMAX=…,TGT=…,OUT=…,RPM=…,KV=…,VBAT=…,T=…,RX=…,CAUSE=…,UP=…,DRT=…`	Response to `F`: state, active fault, last fault and their count, alerts (separated by |), error of the last command, mode (POWER / SPEED), VMAX, target speed, throttle actually sent (‰), speed, KV, VBAT, temperature, telemetry quality, cause of the last state change, seconds since startup, power ceiling in % (100 = no reduction)

4.5 Several ESCs on one link: @ID addressing and RS-485 bus

- **Addressing**: prefix the command with `@<ID>:` (ID from 1 to 31, set in ESC-PLUS Tool). Examples: `@3:A`, `@3:T:+250`, `@3:Q`. The addressed ESC answers with the same prefix: `@3:OK:T:250`, `@3:TEL:…`. The other ESCs do not answer.
- `@0:<command>` addresses **all ESCs** at once, with no response at all (otherwise they would talk at the same time). Example: `@0:D` disarms the whole bus.
- **Bus mode** (“Bus mode (@ID)” option of each ESC): the ESC only processes addressed commands and **no longer sends anything on its own** (neither `BOOT:`, nor `TEL:`, nor `EVT:`): your controller polls each ESC (`@<ID>:Q`) and keeps each setpoint alive (`@<ID>:T:…` at least 10 times per second). Outside bus mode, the direct link is unchanged and addressed commands are also accepted.
- **Taking turns on RS-485** (only one transmitter at a time): one command at a time; wait for the response of the addressed ESC, or failing that a delay (15 ms + the transmission time of the response: count 80 characters, 240 for `F`) before the next command; leave **3 ms of silence** after each response (transceiver turnaround); after `@0:`, wait 15 ms. During a log (`@<ID>:L`), the line belongs to that ESC until `@<ID>:LOG:END`.
- Emergency stop: empty your command queue and send `D` (or `@0:D`) before any other command.
- An echo of your own responses read back on the bus is ignored by the ESC.

4.6 Typical sequence (direct link)

```
← BOOT:ESC_PLUS_V1.2.1            (the ESC is ready, ~5 s after power-up)
← TEL:WAIT,0,0,25,100,2400,9599,0,0,0
→ A
← OK:ARMING
   ... 2 s, high beep ...
← EVT:ARMED
→ T:300                           (then T:300 every 100 ms)
← OK:T:300
← TEL:ARMED,300,2900,26,98,2395,9580,0,0,0
→ T:0                             (stop, motor freewheeling)
← OK:T:0
→ D
← OK:DISARMED
← TEL:WAIT,0,0,26,100,2400,9599,0,0,0
```

If `A` gets no response for 200 ms: resend `A` (rule 3.6).

5. CAN protocol

5.1 Identifiers

Each ESC has an identifier **ID from 1 to 31** (set in ESC-PLUS Tool). Frames use the CAN 2.0A format (11-bit identifiers); extended frames and RTR frames are ignored.

Identifier	Direction	Content	Rate
`0x100 + ID`	Controller → ESC	Command (5.2)	At your own pace; setpoint at least 10 times per second
`0x180 + ID`	ESC → controller	Telemetry (5.3)	10 Hz by default (20 or 50 Hz adjustable), and on QUERY
`0x1C0 + ID`	ESC → controller	Status frame: faults, alerts, error of the last command, VMAX, target speed (5.4)	5 Hz by default (10 Hz or off), and as soon as a fault, an alert or an error changes, and on QUERY
`0x0F0 + g`, `0x0F8 + g`	Controller → ESC	Grouped setpoints for 4 ESCs (5.5), option disabled by default	At least 10 times per second

Example: the ESC with identifier 3 listens to `0x103`, sends its telemetry on `0x183` and its status on `0x1C3`.

5.2 Command frame 0x100 + ID (controller → ESC)

Byte 0	Command	Following bytes	Example for ID 3 (identifier 0x103)
`0x01`	SET_THROTTLE: throttle setpoint	1-2: int16 little-endian, -1000 to +1000 ‰ (out of range: refused)	+300: `01 2C 01`; -500: `01 0C FE`
`0x02`	ARM: arm	-	`02`
`0x03`	DISARM: disarm (immediate stop)	-	`03`
`0x04`	QUERY: immediate telemetry and status frame	-	`04`
`0x05`	IDENT: identification beeps (zero setpoint, motor stopped)	-	`05`
`0x06`	SET_SPEED: speed setpoint (see 3.4)	1-4: int32 little-endian, signed rpm	6000 rpm: `06 70 17 00 00`; -2500 rpm: `06 3C F6 FF FF`

A frame that is too short or an unknown command is refused (error 8 in the status frame). The setpoint is applied only in the Armed state (otherwise error 4, NOTARM).

5.3 Telemetry frame 0x180 + ID (ESC → controller, 8 bytes)

Byte	Content
0	State (code from table 3.1)
1-2	Setpoint in ‰, int16 little-endian (0 outside the armed state). In throttle mode: the setpoint received; in speed mode: the throttle computed by the regulation
3-4	Measured speed in rpm, uint16 little-endian
5	Power stage temperature in °C
6	Internal telemetry quality, 0 to 100%
7	VBAT voltage × 4 (0.25 V steps: 96 = 24.0 V)

5.4 Status frame 0x1C0 + ID (ESC → controller, 8 bytes)

Byte	Content
0	Active fault (table 3.5; 0 = none)
1	Last fault that occurred since power-up (remains displayed after the fault ends)
2	Alerts: bits from table 3.5
3	Bits 0-3: error of the last command (table below); bit 4: speed setpoint active; bit 5: speed limited to VMAX; bit 6: regulation without measured speed; bit 7: identification in progress
4-5	VMAX in rpm, uint16 little-endian (0 = unknown)
6-7	Target speed in rpm, uint16 little-endian (absolute value; 0 with a throttle setpoint)

Code (byte 3, bits 0-3)	Error of the last command
0	None: command applied
1	VMAX: requested speed above VMAX, applied at VMAX
2	CLOFF: speed setpoint not allowed (option to enable in ESC-PLUS Tool)
3	NOKV: KV unknown or VBAT voltage missing
4	NOTARM: setpoint received while the ESC is not armed
5	RANGE: out of range (throttle outside ±1000: refused; negative in unidirectional mode: brought back to 0)
6	FAULT: ARM refused, a fault is active (ARM also acknowledges a blocked propeller)
7	BUSY: ARM refused (startup or motor test in progress)
8	BADFMT: frame too short or unknown command
9	RUNNING: IDENT refused, motor running or non-zero setpoint

Status frame and update

On a new board, the status frame is sent at 5 Hz. On an ESC updated from the very first firmware (1.0.0), it is **off** so as not to add traffic on a bus already in service: enable it in ESC-PLUS Tool (Configuration › CAN › Status frame) if you want the error and fault codes in real time. Without it, the state and the setpoint remain readable in the telemetry `0x180 + ID`.

5.5 Grouped frames: four ESCs per frame (option)

For a controller that drives several thrusters, a single frame carries the setpoints of four ESCs. The “Grouped frame (4 ESCs)” option must be enabled in each ESC (disabled by default: identifiers `0x0F0` to `0x0FF` must be free on your bus).

Identifier	Content (8 bytes)
`0x0F0 + g`	Throttle setpoints of the four ESCs of group g: int16 little-endian each, -1000 to +1000 ‰ (out of range: refused, error 5)
`0x0F8 + g`	Speed setpoints in rpm of the four ESCs of group g: int16 little-endian each (-32,768 to +32,766)

- Group g = (ID - 1) ÷ 4 (integer division), slot = (ID - 1) modulo 4: ESCs 1 to 4 form group 0 (bytes 0-1, 2-3, 4-5, 6-7), ESCs 5 to 8 group 1, … ESCs 29 to 31 group 7.
- The value **`0x7FFF`** (or a frame too short for the slot) means “no setpoint for this ESC”: its setpoint is not kept alive.
- Same handling as an individual setpoint: ESC armed required, kept alive at least 10 times per second, errors in the status frame. Arming, disarming, QUERY and IDENT remain individual (`0x100 + ID`).
- Example - ESC 1 at +300, ESC 2 at -200, ESCs 3 and 4 without setpoint: `0x0F0 [8] 2C 01 38 FF FF 7F FF 7F`.

5.6 Special bus behaviors

- **Identifier conflict**: if the ESC sees a frame `0x180 + its ID` sent by another device, it goes into fault 5 (arming refused) until 10 s after the last foreign frame. Give each ESC a different ID.
- **Bus-off**: the ESC recovers on its own (checked every 500 ms), alert `CAN_BUSOFF` during the incident; the motor is forced to 0 after T1 if the setpoint no longer arrives.
- **Bit rate change**: the new bit rate takes effect after “Write to the ESC” in ESC-PLUS Tool; all devices on the bus must use the same bit rate.

5.7 Typical sequence (ID 3)

```
← 0x183 [8] 01 00 00 00 00 19 64 60     telemetry: state 1 (waiting), 25 °C, 100%, 24.0 V
→ 0x103 [1] 02                          ARM
← 0x183 [8] 02 ...                      state 2 (arming) for 2 s
← 0x183 [8] 03 ...                      state 3 (armed)
→ 0x103 [3] 01 2C 01                    throttle +300, then every 100 ms
← 0x183 [8] 03 2C 01 54 0B 1A 62 5F     armed, setpoint 300, 2900 rpm, 26 °C, 98%, 23.75 V
→ 0x103 [3] 01 00 00                    setpoint 0 (stop, freewheeling)
→ 0x103 [1] 03                          DISARM
← 0x183 [8] 01 ...                      state 1 (waiting)
```

If the state does not change to 2 within 100 ms after ARM: resend ARM (rule 3.6).

6. PWM and analog modes (reminder)

	RC PWM (pin 7)	Analog 0-5 V (pin 8)
Bidirectional	1100 µs full reverse, 1500 µs neutral, 1900 µs full forward (deadband ±20 µs)	0.5 V reverse, 2.5 V neutral, 4.5 V forward (deadband ±0.1 V)
Unidirectional	1100 µs stop → 1900 µs full speed	0.5 V stop → 4.5 V full speed
Arming	Neutral (±50 µs) held for the arming time; the motor only starts after passing through the deadband	Neutral (deadband) held for the arming time
Accepted pulses	500 to 2500 µs, clipped to the set limits; outside this range: invalid signal	Immediate cut-off (failsafe) if the input leaves the range by more than 0.2 V for 100 ms
Signal loss	Motor at 0 after 250 ms (T1), failsafe after 5 s (T2)	Immediate cut-off; re-arming at neutral

PWM installation instruction: set the radio receiver failsafe to “neutral” or to “signal cut”; a receiver that holds the last position is invisible to the ESC.

7. Signaling: LEDs and beeps

The blue “ESP RUN” LED and the motor beeps give the ESC state without any screen (beeps require the power supply connected and the motor stopped).

Situation	LED	Beeps
Startup, valid configuration	On	Beeps 1-2-3, then as many beeps as the mode number (1 PWM, 2 analog, 3 UART, 4 CAN)
Startup, unreadable configuration (default values)	On	Beeps 3-2-1, twice
Waiting without signal / with signal but no ARM	1 Hz	1 low beep / 2 low beeps every 4 s (adjustable: 10 s, 30 s, 60 s, never)
Arming	2.5 Hz	Regular short beeps, then 1 long high beep when armed
Armed	Steady	Silent
Suspended (no setpoint for T1)	5 Hz	Silent
Failsafe	5 Hz	3 fast high beeps every 3 s
Analog input out of range	10 Hz	High-low-high every 3 s
Fault	10 Hz	As many long low beeps as the fault code (2 low or high VBAT, 3 temperature, 4 blocked propeller, 5 ID conflict), every 3 s (10 s for temperature)
Identification	10 Hz for 5 s	Tune 1-3-5-3-1, twice

8. What to know about the motor and the telemetry

- **Motor start**: the first non-zero setpoint starts the motor within 150 to 500 ms (the power stage has an internal delay). At low speed, the telemetry quality (RX % field) normally drops to 55-85%; above about 1300 rpm it exceeds 90%.
- **Stop and reversal**: setpoint 0 = freewheeling (no braking, no overvoltage on the power supply). A reversal waits for the motor to slow down by freewheeling (up to 2 s by default, adjustable from 0 to 5 s) before restarting.
- **Speed**: measured by the power stage (14 poles for the reference motor, adjustable). With a speed setpoint, the target speed returned is the current target along the ramp.
- **Temperature**: that of the power stage, sent about every 1 to 2 s; value confirmed by several measurements; alert `NO_TEMP` if it has been missing for 7 s (common right after startup: not before 15 s).
- **Power derating** (1.2.0): between 70 and 80 °C (by default), the power ceiling drops from 100% to 25% (alert `DERATE`, `DRT=` in `DIAG`); likewise within 1 V above the low VBAT cut-off if it is set. The cut-off itself remains active.
- **Curve, limits and ramps** (1.2.0, neutral by default): the ESC can apply an exponential curve, limit the maximum forward / reverse power and smooth the setpoint with ramp-up / ramp-down times. Everything is set in ESC-PLUS Tool; the safety cut-offs remain immediate.
- **Black box** (1.2.0): the latest events (states, faults, alerts) are kept in flash and readable with ESC-PLUS Tool, useful for support.

9. ESC configuration (ESC-PLUS Tool, over USB)

Connect the ESC to the PC through its USB connector and start ESC-PLUS Tool: the ESC is detected automatically. Adjust the settings, then click “Write to the ESC”. Settings that concern integration:

Setting	Where	Default value	To do
Command mode	Configuration	PWM	Choose UART or CAN
ID	Configuration	1	A different number for each ESC on the same bus (1 to 31)
Direction, Rotation	Configuration	Bidirectional, Normal	Unidirectional if the thruster only turns one way; Reversed to swap forward and reverse without rewiring
Motor KV, Motor poles	Configuration › Motor	0 (unknown), 14	Enter the KV for the speed setpoint and VMAX
UART baud rate, TEL telemetry	Configuration › UART	115,200 baud, 10 Hz	According to your controller
Bus mode (@ID)	Advanced settings › UART	Off	Enable for an RS-485 bus
CAN bit rate, telemetry, status frame	Configuration › CAN	500 kbit/s, 10 Hz, 5 Hz	Bus bit rate; status frame to be enabled after an update from 1.0.0
Grouped frame (4 ESCs)	Advanced settings › CAN	Off	Enable if you use `0x0F0` / `0x0F8`
Speed command (UART, CAN)	Advanced settings › Closed loop	Off	Enable for `S:` / `0x06`; set VMAX (%) and the gains on your thruster
Arming time, T1, T2	Safety, Advanced settings › Safety	2 s, 500 ms, 5 s	Keep the default values unless there is a specific need
Low / high VBAT cut-off, Temperature cut-off	Safety	Off, 80 °C	Set the VBAT thresholds according to the battery (6 V minimum)
Beep while waiting	Safety	Every 4 s	10 s, 30 s, 60 s or never

An ESC is updated through the same USB cable (Update tab): the configuration is kept. A new firmware always remains compatible with controllers written for a previous version.

10. Quick reference

I want to...	UART	CAN (ID = n)
Know whether the ESC is ready	Line `BOOT:` received, or `Q` → `TEL:WAIT`, …	Telemetry `0x180+n` with byte 0 = 1
Arm	`A` → `OK:ARMING` → `EVT:ARMED`	`0x100+n [1] 02` → telemetry byte 0 = 2 then 3
Go forward at 30%	`T:300` every 100 ms	`0x100+n [3] 01 2C 01` every 100 ms
Go in reverse at 50%	`T:-500` every 100 ms	`0x100+n [3] 01 0C FE` every 100 ms
Hold 3000 rpm	`S:3000` every 100 ms (option enabled, KV set)	`0x100+n [5] 06 B8 0B 00 00` every 100 ms
Stop the motor	`T:0`	`0x100+n [3] 01 00 00`
Disarm (immediate stop)	`D` → `OK:DISARMED`	`0x100+n [1] 03`
Read the state and faults	Fields 1, 9 and 10 of `TEL:`; `F` for details	Telemetry byte 0; bytes 0 to 3 of the status frame `0x1C0+n`
Identify a thruster (beeps)	`I`	`0x100+n [1] 05`
After a failsafe (state 4)	`A` again	`0x100+n [1] 02` again
Command without response	Resend after 200 ms	Check the telemetry, resend