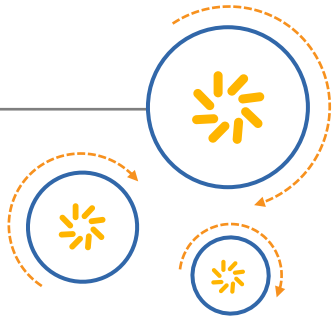
Qualcomm Technologies, Inc.



Qualcomm Snapdragon Navigator ESC

Protocol Specification

80-P4698-19 B

August 23, 2017

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	August 2017	Initial release
B	August 2017	Updated Section 4.8. Added Sections 4.8.2 and 4.8.3.

Contents

1 Introduction

1.1 Purpose

This document provides the definition of the UART protocol for the Qualcomm® Snapdragon Navigator™ Electronic Speed Controller (ESC).

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 Principles of operation

2.1 Supported devices

- Qualcomm Snapdragon Navigator ESC: 20-H9598-H1, 20-H9875-H1

2.2 Power on sequence

When power is applied or ESC is reset via software, the bootloader will start and will perform the following:

1. Wait for a command from the host for firmware update. The status LEDs on all ESCs will blink rapidly during this time.
2. If no firmware update is initiated in 1 second, the bootloader will verify the integrity of the firmware and begin execution.
3. If the firmware is corrupt, the ESC will remain in bootloader mode, ready to accept new firmware.
4. When ESC firmware starts, the status LED will flash once and a short tone will be generated by the attached motor. (Note: this behavior is optional.)

2.3 Sending power and RPM commands

- The RPM command is the default method used for flight. It also includes RGB LED control for LEDs attached to the ESCs.
- Spinning the motor in RPM mode without propellers will cause motor RPM oscillations and audible performance differences due to the absence of load on the motor. Although no permanent damage should be done to ESC/motor in this mode, it is not recommended to spin the motors in RPM mode without propellers.
- The Power command can be used for testing the ESC/motor without the propeller. When a fixed power is applied, all motors with the same propeller should spin at similar RPMs. This command can be used in production testing.
- For safety, from non-spinning state, it is required to send at least 5 non-zero Power or RPM commands in a row (with 2 ms intervals) before the ESC will start the motor.
- For safety, from spinning state, it is required to send at least 10 zero Power or RPM commands in a row (with 2 ms intervals) before the ESC will stop the motor.
- If ESC is spinning and the Power or RPM commands stop, it will time out and stop spinning in ~0.3 seconds.
- When the ESC is spinning a motor and is commanded to stop or it times out, the default behavior is to apply full brake, which ensures the fastest stopping of the propeller. (Note: this behavior is configurable.)

- If self-tightening propellers are used, applying full brake during motor stopping can cause the propeller to unscrew and fly off. Braking behavior should be modified for those via ESC configuration (instructions will be provided in a future revision of this document or as needed).

2.4 Requesting feedback from ESCs

- The only way to request feedback is using the Power or RPM commands. See details in the description for the RPM and Power commands.
- Do not request feedback from more than one ESC at a time; feedback will be sent instantaneously, so if multiple ESCs transmit on the same line, packet errors will occur.

2.5 Generating audible tones using motors

- When the motor is not spinning, ESC can generate a range of tones in order to tell user vehicle status. See command 3 for more details.

2.6 Detecting and recovering from Stall Condition

- For protection, ESC firmware has stall detection and will stop motor operation if the motor rpm is determined to be close to zero, or if one or more phases become disconnected. Even if the non-zero RPM command continues to be sent to ESC in this state, it will not restart spinning for safety reasons.
- In “stall condition,” the ESC will report state 0 and RPM 0 (just like when it is simply stopped).
- In order to reset this “stall condition,” ESC power can be cycled, or a reset sequence can be sent from the host.
- The sequence for resetting “stall condition” is sending 10 packets in a row with 0 Power or 0 RPM commands (it suggested to send 20 packets).

3 Generic packet format

3.1 Packet format

Start Byte	Total Length	Type	Payload	Checksum (CRC16)
0xAF	1 Byte	1 byte	Up to 250 Bytes	2 Bytes

3.2 Description of the fields

Start Byte	Common byte for all packets; used for synchronization
Total length	Length of the whole packet, including all bytes from Start Byte to Checksum (inclusive)
Type	1-byte packet identifier
Payload	Variable length payload (up to 250 bytes, depending on the packet type)
CRC16	Checksum of the packet (all bytes excluding Start Byte and Checksum itself)

3.3 Notes

- Default baud rate for ESC communication is 250,000 baud.
- Bit format is 8N1 (8 data bits, no parity, 1 stop bit).
- Minimum packet size is 5 bytes (payload size is zero).
- Checksum: includes all bytes starting from Total Packet Length and ends with last byte of the payload (inclusive).
- Maximum inter-character delay is 800us at 250,000 baud. If a larger delay is detected during transmission of the packet, the packet will be considered invalid and will be ignored.
- Feedback from ESCs is requested by sending a PWR or RPM command (see command descriptions).

4 Packet definitions

4.1 Version request packet

- Packet type = 0
- This packet is only available for firmware version 1 or later (version 0 will not respond to this request).

Name	Bytes	Hex	Decimal	Description
Start Byte	1U	0xAF	175	Start byte common to all packets
Length	1U	6	6	Total length of the packet
Type	1U	0	0	Packet type for version request
ID	1U	0	0	ID of the ESC that is being queried
Checksum	2U	0x91, 0xC1	145, 193	CRC16

4.2 Version response packet

- Packet Type = 109
- This packet is sent from ESC after it receives a Version Request Packet.

Packet format and example

- Response from ESC ID 0, SW Version is 123, HW version is 456, unique MCU is 123456

```
DEC: 175, 14, 109, 0, 123, 0, 200, 1,  64, 226, 1, 0, 127, 49
HEX: 0xAF, 0x0E, 0x6D, 0x00, 0x7B, 0x00, 0xC8, 0x01, 0x40, 0xE2, 0x01,
0x00, 0x7F, 0x31
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1U	0xAF	175	Start byte common to all packets
Length	1U	0x0E	14	Total length of the packet
Type	1U	0x6D	109	Packet type for Feedback Response
ID	1U	0x00	0	ID of the ESC that is replying (0..7)
SW Version	2U	0x7B, 0x00	123, 0	Software version number (123 in this case)
HW Version	2U	0xC8, 0x01	200, 1	Hardware version number (456 in this case)
Unique ID	4U	0x40,0xE2,1,0	64, 226, 1, 0	Unique ID of the MCU (if available) (123456)
Checksum	2U	0x7F, 0x31	127, 49	CRC16

4.3 Open-loop control packet – Power control

- Packet Type = 1
- Constant PWM duty cycle is applied once the motor has started.
- This is called open-loop control – RPM is not controlled.
- This packet contains Power setting for all four ESCs.
- LED outputs are set in the same command (RGB led on each ESC).
- Feedback can be requested from any ESC (one at a time).
- In order to request feedback from ESC, set the LSB of the Power command to 1. If feedback is not requested, set LSB of the Power command to 0.
- Do not request feedback from multiple ESCs at the same time, since this will cause errors on UART TX line due to simultaneous transmission. In other words, out of the four Power commands, at most one of them should have LSB=1.
- Feedback will be transmitted from ESC as soon as possible.
- Format of the feedback is defined in Packet 128.
- ESC LED information is included into the control packet for convenience.
- There is no ACK packet from ESCs for this command.

Packet format and example

Send PWR command 80 (10%) to all ESCs and request feedback from ESC0; all LEDs ON

```
DEC: 175 15 1 81 0 80 0 80 0 80 0 255 15 63 246
HEX: 0xAF 0x0F 0x01 0x51 0x00 0x50 0x00 0x50 0x00 0x50 0x00 0xFF 0x0F 0x3F
0xF6
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1 U	0xAF	175	Start byte common to all packets
Length	1 U	0x0F	15	Total length of the packet
Type	1 U	1	1	Packet type for power control
Power0	2S	0x51, 0x00	81, 0	Power command for ESC0. [-800:800] range. 800 means 100% duty cycle. Negative values means reverse direction. LSB of this value specifies whether feedback from this ESC should be sent back to Host.
Power1	2S	0x50, 0x00	80, 0	See description for Power0
Power2	2S	0x50, 0x00	80, 0	See description for Power0
Power3	2S	0x50, 0x00	80, 0	See description for Power0
LED byte 1	1U	0xFF	255	bit0 = ESC0 Red, bit1 = ESC0, Green, bit2 = ESC0 Blue, bit3 = ESC1 Red, bit4 = ESC1 Green, bit5 = ESC1 Blue, bit6 = ESC2 Red, bit7 = ESC2 Green
LED byte 2	1U	0x0F	15	bit0 = ESC2 Blue, bit1 = ESC3 Red, bit2 = ESC3 Green, bit3 = ESC3 Blue, bits 4:7 = unused
Checksum	2U	0x3F, 0xF6	63, 246	CRC16

4.4 Closed-loop control packet – RPM control

- Packet Type = 2
- In this mode, performance has been tuned to achieve target RPM optimally.
- This packet contains RPM setting for all four ESCs.
- LED outputs are set in the same command (RGB led on each ESC).
- Feedback can be requested from any ESC (one at a time).
- In order to request feedback from ESC, set the LSB of the RPM command to 1. If feedback is not requested, set LSB of the RPM command to 0.
- Do not request feedback from multiple ESCs at the same time, since this will cause errors on UART TX line due to simultaneous transmission. In other words, out of the four Power commands, at most one of them should have LSB=1
- Feedback will be transmitted from ESC as soon as this packet has been processed.
- Format of the feedback is defined in Packet 128.
- ESC LED information is included into the control packet for convenience.
- For proper operation, this command should be sent at 500Hz.
- There is no ACK packet from ESCs for this command.

Packet format and example

Send RPM command 7000 to all ESCs and request feedback from ESC1, all LEDs ON

```
DEC: 175 15 2 88 27 89 27 88 27 88 27 255 15 34 43
HEX: 0xAF 0x0F 0x02 0x58 0x1B 0x59 0x1B 0x58 0x1B 0x58 0x1B 0xFF 0x0F 0x22
0x2B
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1 U	0xAF	175	Start byte common to all packets
Length	1 U	0x0F	15	Total length of the packet
Type	1 U	2	2	Packet type for power control
RPM0	2S	0x58, 0x1B	88, 27	RPM command for ESC0. Range is dependent on motor/propeller. Negative values means reverse direction. LSB of this value specifies whether feedback from this ESC should be sent back to Host.
RPM1	2S	0x59, 0x1B	89, 27	See description for RPM0
RPM2	2S	0x58, 0x1B	88, 27	See description for RPM0
RPM3	2S	0x58, 0x1B	88, 27	See description for RPM0
LED byte 1	1U	0xFF	255	bit0 = ESC0 Red, bit1 = ESC0, Green, bit2 = ESC0 Blue, bit3 = ESC1 Red, bit4 = ESC1 Green, bit5 = ESC1 Blue, bit6 = ESC2 Red, bit7 = ESC2 Green
LED byte 2	1U	0x0F	15	bit0 = ESC2 Blue, bit1 = ESC3 Red, bit2 = ESC3 Green, bit3 = ESC3 Blue, bits 4:7 = unused
Checksum	2U	0x22, 0x2B	34, 43	CRC16

4.5 Tone generation packet

- Packet Type = 3
- Tone frequency, duration and loudness can be controlled.
- All UART commands are ignored while tone is generated.

Packet format and example

All ESCs produce a tone with period=30, duration=5, power=20, mask=255

```
DEC: 175 9 3 30 5 20 255 29 235
HEX: 0xAF 0x09 0x03 0x1E 0x05 0x14 0xFF 0x1D 0xEB
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1U	0xAF	175	Start byte common to all packets
Length	1U	0x09	9	Total length of the packet
Type	1U	0x03	3	Packet type for tone generation
Period	1U	0x1E	30	Range [0 255] is the period of the sound signal (inverse of frequency)
Duration	1U	0x05	5	Range [0..255] is duration of the sound (1LSB = 13ms)
Power	1U	0x14	20	Range [0..100] is relative loudness of the sound
Mask	1U	0xFF	255	Bit mask that represents which ESC should execute the sound. bit0=1 : ESC0 should generate sound, etc
Checksum	2U	0x1D, 0xEB	29, 235	CRC16

4.6 LED control packet (standalone)

- Packet Type = 5
- Normally LEDs should be controlled via the Power and RPM packet in order to avoid sending two packets every update cycle; however, this packet is provided for convenience.

Packet format and example

```
LED bit format: [R0 G0 B0  R1 G1 B1  R2 G2 B2  R3 G3 B3]
LED bit format example: [1 0 0  0 1 0  0 0 1  1 1 1]
```

- This means ESC0 should turn on Red LED, ESC1 should turn on Green LED, ESC2 should turn on Blue LED, ESC3 should turn on RGB (White).

```
DEC: 175 7 5 17 15 93 5
HEX: 0xAF 0x07 0x05 0x11 0x0F 0x5D 0x05
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1 U	0xAF	175	
Length	1 U	0x07	7	Total length of the packet
Type	1 U	0x05	5	Packet type for LED control

Name	Bytes	Hex	Decimal	Description
LED byte 1	1U	0x11	17	bit0 = ESC0 Red, bit1 = ESC0, Green, bit2 = ESC0 Blue, bit3 = ESC1 Red, bit4 = ESC1 Green, bit5 = ESC1 Blue, bit6 = ESC2 Red, bit7 = ESC2 Green
LED byte 2	1U	0x0F	15	bit0 = ESC2 Blue, bit1 = ESC3 Red, bit2 = ESC3 Green, bit3 = ESC3 Blue, bits 4:7 = unused
Checksum	2U	0x5D, 0x05	93, 5	CRC16

4.7 Reset request packet

- Packet Type = 10
- Can be used to perform software reset of the ESC.
- Is used primarily for entering bootloader for firmware update.
- This command is only accepted if the motor is not spinning.

Packet format and example

```
Send Reset to ESC0
DEC: 175 11 10 82 69 83 69 84 48 85 128
HEX: 0xAF 0x0B 0x0A 0x52 0x45 0x53 0x45 0x54 0x30 0x55 0x80
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1U	0xAF	175	Start byte common to all packets
Length	1U	0x0B	11	Total length of the packet
Type	1U	0x0A	10	Packet type for Reset Request
byte0	1U	0x52	82	'R'
byte1	1U	0x45	69	'E'
byte2	1U	0x53	83	'S'
byte3	1U	0x45	69	'E'
byte4	1U	0x54	84	'T'
byte5	1U	0x30	48	'0' - ASCII number of the ESC to be reset (0..3)
Checksum	2U	0x55, 0x80	85, 128	CRC16

4.8 Feedback response packets

NOTE: Numerous changes were made in this section.

- Feedback response packet versions vary by firmware version as follows:
  - Firmware versions prior to version 4 support response packet v1 (see Section 4.8.1)
  - Firmware versions 4-19 support response packet v2 (see Section 4.8.2)
  - Firmware versions 20+ support response packets v2 and v3 (see Section 4.8.3)
- Packet type = 128
- This packet is sent from ESC after it has been requested using a Power or RPM command

- State supports the following values:
  - 0 – Stopped
  - 4 – Spinning up
  - 5 – Spinning forward
  - 6 – Spinning in reverse
  - 10 – Stall condition

NOTE: State only reports 6 if a negative RPM is commanded. Default directions are assigned based on the ESC ID (two motors spin forward and two spin backward), but all report state 5 if commanded positive RPM/Power.

4.8.1 Feedback response packet v1

Packet format and example

```
DEC: 175   11  128    5    4   55  148   10  228   56  150
id=0, state=5, counter=148, voltage=8.176, rpm=14084
```

Name	Bytes	Hex	Decimal	Description
Start byte	1U	0xAF	175	Start byte common to all packets
Length	1U	0x0B	11	Total length of the packet
Type	1U	0x80	128	Packet type for feedback response
ID and State	1U	0x05	5	Bits 0:3 = State Bits 4:7 = ID
RPM	2U	0x04, 0x37	4, 55	Current RPM of the motor (14084 RPM)
Cmd counter	1U	0x94	148	Number of commands received by ESC (148)
PWM duty cycle	1S	0x0A	10	Applied PWM duty cycle (10). ▪ Range is -100 to 100 ▪ A negative value means braking ▪ 100 is maximum power
Voltage	1S	0xE4	228 (-28)	Voltage measured by the specific ESC. Voltage = (-28)/34.0 + 9.0 = 8.176 V
Checksum	2U	0x38, 0x96	56, 150	CRC16

4.8.2 Feedback response packet v2

NOTE: This section was added to this document revision.

Packet format and example

```
DEC: 175 12 128 5 82 41 8 30 194 48 97 246
id=0, state=5, counter=8, pwr=+030, rpm=10578, voltage=12.482V
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1U	0xAF	175	Start byte common to all packets
Length	1U	0x0B	12	Total length of the packet
Type	1U	0x80	128	Packet type for feedback response
ID and State	1U	0x05	5	Bits 0:3 = State Bits 4:7 = ID
RPM	2U	0x52, 0x29	82, 41	Current RPM of the motor (10578 RPM)
Cmd counter	1U	0x08	8	Number of commands received by ESC (8)
PWM duty cycle	1S	0x1E	30	Applied PWM duty cycle (30). ▪ Range is -100 to 100 ▪ A negative value means braking ▪ 100 is maximum power
Voltage	2U	0xC2, 0x30	194, 48	Voltage (in millivolts) measured by the ESC. Voltage = 12.482V
Checksum	2U	0x61, 0xF6	97, 246	CRC16

4.8.3 Feedback response packet v3

NOTE: This section was added to this document revision.

Packet format and example

```
DEC: 175 16 128 5 164 17 99 30 133 47 89 0 199 12 155 8

id=0, state=5, counter=99, pwr=+030, rpm=4516, voltage=12.165V,
current=0.712A, temperature=32.710C
```

Name	Bytes	Hex	Decimal	Description
Start Byte	1U	0xAF	175	Start byte common to all packets
Length	1U	0x0B	11	Total length of the packet
Type	1U	0x80	128	Packet type for feedback response
ID + State	1U	0x05	5	Bits 0:3 = State Bits 4:7 = ID
RPM	2U	0xA4, 0x11	164, 17	Current RPM of the motor (4516 RPM)
Cmd Counter	1U	0x63	99	Number of commands received by ESC (99)
PWM Duty Cycle	1S	0x1E	30	Applied PWM duty cycle (30). ▪ Range is -100 to 100 ▪ A negative value means braking 100 is maximum power
Voltage	2U	0x85, 0x2F	133, 47	Voltage (in millivolts) measured by the ESC Voltage = 12.165 V
Current	2U	0x59, 0x00	89, 0	Current (in 8mA res.) measured by the ESC = 0.712A
Temperature	2U	0xC7, 0x0C	199, 12	Temperature of the MCU in degrees C * 100 = 32.71
Checksum	2U	0x9B, 0x08	155, 8	CRC16

A Command examples

A.1 Open loop (PWR/PWM) command examples

Send PWR command 0 to all ESCs and request feedback from ESC0; all LEDs ON

```
DEC: 175 15 1 1 0 0 0 0 0 0 0 255 15 36 219
HEX: 0xAF 0x0F 0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0x0F 0x24
0xDB
```

Send PWR command 0 to all ESCs and request feedback from ESC1; all LEDs ON

```
DEC: 175 15 1 0 0 1 0 0 0 0 0 255 15 180 210
HEX: 0xAF 0x0F 0x01 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0xFF 0x0F 0xB4
0xD2
```

Send PWR command 0 to all ESCs and request feedback from ESC2; all LEDs ON

```
DEC: 175 15 1 0 0 0 0 1 0 0 0 255 15 116 207
HEX: 0xAF 0x0F 0x01 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0xFF 0x0F 0x74
0xCF
```

Send PWR command 0 to all ESCs and request feedback from ESC3; all LEDs ON

```
DEC: 175 15 1 0 0 0 0 0 0 1 0 255 15 116 226
HEX: 0xAF 0x0F 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0xFF 0x0F 0x74
0xE2
```

Send PWR command 80 (10%) to all ESCs and request feedback from ESC0; all LEDs ON

```
DEC: 175 15 1 81 0 80 0 80 0 80 0 255 15 63 246
HEX: 0xAF 0x0F 0x01 0x51 0x00 0x50 0x00 0x50 0x00 0x50 0x00 0xFF 0x0F 0x3F
0xF6
```

Send PWR command 80 (10%) to all ESCs and request feedback from ESC1; all LEDs ON

```
DEC: 175 15 1 80 0 81 0 80 0 80 0 255 15 175 255
HEX: 0xAF 0x0F 0x01 0x50 0x00 0x51 0x00 0x50 0x00 0x50 0x00 0xFF 0x0F 0xAF
0xFF
```

Send PWR command 80 (10%) to all ESCs and request feedback from ESC2; all LEDs ON

```
DEC: 175 15 1 80 0 80 0 81 0 80 0 255 15 111 226
HEX: 0xAF 0x0F 0x01 0x50 0x00 0x50 0x00 0x51 0x00 0x50 0x00 0xFF 0x0F 0x6F
0xE2
```

Send PWR command 80 (10%) to all ESCs and request feedback from ESC3; all LEDs ON

```
DEC: 175 15 1 80 0 80 0 80 0 81 0 255 15 111 207
HEX: 0xAF 0x0F 0x01 0x50 0x00 0x50 0x00 0x50 0x00 0x51 0x00 0xFF 0x0F 0x6F
0xCF
```

A.2 Closed loop (RPM) command examples

Send RPM command 0 to all ESCs and request feedback from ESC0, all LEDs ON

```
DEC: 175 15 2 1 0 0 0 0 0 0 0 255 15 43 159
HEX: 0xAF 0x0F 0x02 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0x0F 0x2B
0x9F
```

Send RPM command 0 to all ESCs and request feedback from ESC1, all LEDs ON

```
DEC: 175 15 2 0 0 1 0 0 0 0 0 255 15 187 150
HEX: 0xAF 0x0F 0x02 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0xFF 0x0F 0xBB
0x96
```

Send RPM command 0 to all ESCs and request feedback from ESC2, all LEDs ON

```
DEC: 175 15 2 0 0 0 0 1 0 0 0 255 15 123 139
HEX: 0xAF 0x0F 0x02 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0xFF 0x0F 0x7B
0x8B
```

Send RPM command 0 to all ESCs and request feedback from ESC3, all LEDs ON

```
DEC: 175 15 2 0 0 0 0 0 0 1 0 255 15 123 166
HEX: 0xAF 0x0F 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0xFF 0x0F 0x7B
0xA6
```

Send RPM command 7000 to all ESCs and request feedback from ESC0, all LEDs ON

```
DEC: 175 15 2 89 27 88 27 88 27 88 27 255 15 178 34
HEX: 0xAF 0x0F 0x02 0x59 0x1B 0x58 0x1B 0x58 0x1B 0x58 0x1B 0xFF 0x0F 0xB2
0x22
```

Send RPM command 7000 to all ESCs and request feedback from ESC1, all LEDs ON

```
DEC: 175 15 2 88 27 89 27 88 27 88 27 255 15 34 43
HEX: 0xAF 0x0F 0x02 0x58 0x1B 0x59 0x1B 0x58 0x1B 0x58 0x1B 0xFF 0x0F 0x22
0x2B
```

Send RPM command 7000 to all ESCs and request feedback from ESC2, all LEDs ON

```
DEC: 175 15 2 88 27 88 27 89 27 88 27 255 15 226 54
HEX: 0xAF 0x0F 0x02 0x58 0x1B 0x58 0x1B 0x59 0x1B 0x58 0x1B 0xFF 0x0F 0xE2
0x36
```

Send RPM command 7000 to all ESCs and request feedback from ESC3, all LEDs ON

```
DEC: 175 15 2 88 27 88 27 88 27 89 27 255 15 226 27
HEX: 0xAF 0x0F 0x02 0x58 0x1B 0x58 0x1B 0x58 0x1B 0x59 0x1B 0xFF 0x0F 0xE2
0x1B
```

A.3 LED control examples (standalone)

```
LED state: [1 0 0  0 1 0  0 0 1  1 1 1]
DEC: 175 7 5 17 15 93 5
HEX: 0xAF 0x07 0x05 0x11 0x0F 0x5D 0x05

LED state: [1 0 0  1 0 0  1 0 0  1 0 0]
DEC: 175 7 5 73 2 167 0
HEX: 0xAF 0x07 0x05 0x49 0x02 0xA7 0x00

LED state:  [0 1 0  0 1 0  0 1 0  0 1 0]
DEC: 175 7 5 146 4 125 242
HEX: 0xAF 0x07 0x05 0x92 0x04 0x7D 0xF2

LED state: [0 0 1  0 0 1  0 0 1  0 0 1]
DEC: 175 7 5 36 9 202 87
HEX: 0xAF 0x07 0x05 0x24 0x09 0xCA 0x57
```

A.4 Reset command examples

Send Reset to ESC0

```
DEC: 175 11 10 82 69 83 69 84 48 85 128
HEX: 0xAF 0x0B 0x0A 0x52 0x45 0x53 0x45 0x54 0x30 0x55 0x80
```

Send Reset to ESC 1

```
DEC: 175 11 10 82 69 83 69 84 49 148 64
HEX: 0xAF 0x0B 0x0A 0x52 0x45 0x53 0x45 0x54 0x31 0x94 0x40
```

Send Reset to ESC 2

```
DEC: 175 11 10 82 69 83 69 84 50 212 65
HEX: 0xAF 0x0B 0x0A 0x52 0x45 0x53 0x45 0x54 0x32 0xD4 0x41
```

Send Reset to ESC 3

```
DEC: 175 11 10 82 69 83 69 84 51 21 129
HEX: 0xAF 0x0B 0x0A 0x52 0x45 0x53 0x45 0x54 0x33 0x15 0x81
```

A.5 Feedback packet examples

```
DEC: 175   11  128    5    4   55  148   10  228   56  150
id=0, state=5, counter=148, voltage=8.176, rpm=14084

DEC: 175   11  128   21  212   55   66   10  222  154   63
id=1, state=5, counter=66, voltage=8.000, rpm=14292

DEC: 175   11  128   37  120   56   67   10  222   93    3
id=2, state  5, counter=67, voltage=8.000, rpm=14456

DEC: 175   11  128   53  112   54  136   10  159   13  116
id=3, state  5, counter=136, voltage=6.147, rpm=13936
```

B CRC16 function implementation

```
#ifndef CRC16_H
#define CRC16_H

#include <stdint.h>

//CRC lookup table for bytes, generating polynomial is 0x8005
//initial value should be 0xFFFF
//usage : uint16_t crc_val = crc16(0xFFFF, buffer, length);

const uint16_t crc16_table[256] =
{
  0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241,
  0xc601, 0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0xc481, 0x0440,
  0xcc01, 0x0cc0, 0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40,
  0x0a00, 0xcac1, 0xcb81, 0x0b40, 0xc901, 0x09c0, 0x0880, 0xc841,
  0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40,
  0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,
  0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,
  0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,
  0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,
  0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,
  0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,
  0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,
  0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,
  0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40,
  0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,
  0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,
  0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,
  0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
  0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,
  0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,
  0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,
  0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,
  0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640,
  0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041,
  0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241,
  0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,
  0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,
  0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841,
```

```
  0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,
  0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,
  0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,
  0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040,
};

static inline uint16_t crc16_byte(uint16_t crc, const uint8_t c)
{
  uint8_t lut = (crc ^ c) & 0xFF;
  return (crc >> 8) ^ crc16_table[lut];
}

uint16_t crc16(uint16_t crc, uint8_t const *buffer, uint16_t len)
{
  while (len--)
    crc = crc16_byte(crc, *buffer++);
  return crc;
}


#endif
```

C Terms and conditions

THIS TERMS AND CONDITIONS OF USE (THE "AGREEMENT") IS A LEGALLY BINDING AGREEMENT BETWEEN QUALCOMM TECHNOLOGIES, INC. ("QTI") AND YOU OR THE LEGAL ENTITY YOU REPRESENT ("You" or "Your"). QTI IS WILLING TO PROVIDE THESE INSTRUCTION SETS AND ANY ASSOCIATED DOCUMENTATION (COLLECTIVELY REFERRED TO AS THE "INSTRUCTIONS") TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT AND AGREE TO ALL OF THE TERMS AND CONDITIONS IN THIS AGREEMENT. BY DOWNLOADING AND USING THE INSTRUCTIONS YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THESE TERMS, QTI IS UNWILLING TO AND DOES NOT LICENSE THE INSTRUCTIONS TO YOU. IF YOU DO NOT AGREE TO THESE TERMS YOU MUST DISCONTINUE THE USE OF THE INSTRUCTIONS AND YOU SHALL NOT USE THE INSTRUCTIONS OR RETAIN ANY COPIES OF THE INSTRUCTIONS. ANY USE OR POSSESSION OF THE INSTRUCTIONS BY YOU IS SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.
BY USING THE INSTRUCTIONS, YOU REPRESENT, WARRANT AND CERTIFY THAT: YOU ARE AN AUTHORIZED REPRESENTATIVE OF THE LEGAL ENTITY YOU REPRESENT; YOU HAVE READ THIS AGREEMENT AND UNDERSTAND IT, INCLUDING THE CIVIL CODE SECTION BELOW; YOU HAVE THE AUTHORITY TO BIND THE LEGAL ENTITY YOU REPRESENT TO THE TERMS AND CONDITIONS OF THIS AGREEMENT; AND YOU AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS.

1. **DESIGN OF YOUR PRODUCTS.** You acknowledge that QTI has had and will have no participation in or control over – and no responsibility for – the design or assembly of Your products, including drones, the integration of any of the attached Materials into Your products, including drones, or the sale or marketing of Your products, including drones.
2. **ASSUMPTION OF RISK.** You acknowledge that the operation of the Materials, alone or in a product, including drones, is a potentially dangerous activity and may result in significant harm to property or injury or death to persons. You agree to include on Your products prominent warnings of such risks, as may be required by law or regulation and as may be necessary or prudent to advise users of such risks. You, and not QTI, assume all risks and liabilities that may result from the use of the Materials, whether or not modified by You and whether or not implemented in connection with a reference design provided by QTI or any of its Affiliates.
3. **SAFETY PRECAUTIONS AND PROCEDURES.** You will operate Your products, including drones, in compliance with any and all safety procedures and precautions as are reasonable for the operation of Your products, including drones. This may include using blade guards on drones or operating any drone sufficiently far away from people, property or other hazardous structures e.g., electricity lines. In addition, You will operate Your products, including drones, in compliance with all applicable laws and regulations, including safety and operational guidelines, that apply to the use of Your products, including drones.
4. **WARRANTY**. THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAWS, QTI AND ITS AFFILAITES EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, TITLE, FITNESS FOR A PARTICULAR PURPOSE AND WARRANTIES THAT THE MATERIALS ARE FREE FROM THE RIGHTFUL CLAIM OF ANY THIRD PARTY, BY WAY OF INFRINGEMENT OR THE LIKE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY QTI OR ITS AUTHORIZED REPRESENTATIVES SHALL CREATE OR EXTEND ANY WARRANTY.
5. **LIMITATION OF LIABILITY.** IN NO EVENT SHALL QTI OR ANY OF ITS AFFILIATES BE LIABLE TO YOU FOR ANY INCIDENTAL, CONSEQUENTIAL OR SPECIAL DAMAGES, INCLUDING, BUT NOT LIMITED TO, ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT OR THE USE OR INABILITY TO USE, OR THE DELIVERY OF OR FAILURE TO DELIVER THE MATERIALS EVEN IF QTI OR ITS AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. QTI'S TOTAL, CUMULATIVE LIABILITY FOR DIRECT DAMAGES ARISING FROM OR IN CONNECTION WITH THIS AGREEMENT, WILL BE LIMITED TO A TOTAL AMOUNT OF ONE HUNDRED UNITED STATES DOLLARS (US $100). MULTIPLE CLAIMS WILL BE AGGREGATED TO DETERMINE THE SATISFACTION OF THIS LIMIT.
6. **INDEMNITY.** You agree to defend, indemnify and hold QTI, its Affiliates, employees, directors, agents, licensors, successors and assignees (each an "Indemnified Party") harmless from any and all claims, penalties, demands, causes of action, liabilities, lawsuits, or damages, including attorneys' fees and costs, that result from or relate to the Materials or any product, including drones, made, used, sold, imported, exported, or distributed by You which uses the Materials or any part or derivative work thereof, even where such product uses the Materials without modification and even where the design of such product is identical to the design of any reference product, including drones. This indemnification includes, without limitation, any claims for damages to property or injury or death to persons and any investigation, enforcement action, civil penalty, or other action conducted or cost imposed by the United States Federal Aviation Administration (FAA) or any governmental entity of the United States or any other government. If any third party asserts a claim or initiates an action against an Indemnified Party for which You are responsible under this Section, QTI shall promptly notify You when it becomes aware of such claim or action, provided, however, that any delay in notification shall not relieve You from Your indemnification obligations under this Agreement. QTI shall have the right to participate in the defense of such claim or action, including any related settlement negotiations. No such claim or action may be settled or compromised without QTI's express written consent, which may be conditioned upon the execution of a release of all claims against the Indemnified Parties by the party bringing such claim or action.
7. **GENERAL RELEASE.**
   a. Release. In consideration of QTI's allowing You to use the Materials, You on behalf of Yourself, Your, affiliates, agents, successors and assigns, hereby fully and forever release and discharge QTI (and all of its officers, directors, employees, agents, successors, assigns, control persons, subsidiaries and Affiliated companies, together the "Released Parties"), from any and all claims, demands, liabilities, obligations, responsibilities, suits, actions and causes of action against any of the Released Parties, whether liquidated or unliquidated, fixed or contingent, known or unknown, presently existing or which, through the passage of time, might arise in the future, related to or arising out of Your use of the Materials.
   b. Waiver and Acknowledgement. You hereby expressly waives all rights under Section 1542 of the Civil Code of the State of California, and under any and all similar laws of any governmental entity. You hereby confirm that you are aware that said Section 1542 of the Civil Code provides as follows:
   "A general release does not extend to claims which the creditor does not know or suspect to exist in his favor at the time of executing the release, which if known by him must have materially affected his settlement with the debtor."
8. **INSURANCE:** If You make or distribute drones, or participate in making or distributing drones, You agree to and will maintain (throughout the time You are using any of the Materials in connection with such drones) insurance providing adequate coverage for potential product liability, personal injury, property damage, and privacy claims and litigation associated with such drones and/or Materials.