KISS ESC 32-bit series onewire telemetry protocol

The KISS 32-bit series supports a onewire telemetry protocol. You will get:

- Temperature (resolution 1°C)
- Voltage (resolution 0.01V)
- Current (resolution 0.01A)
- Consumption (resolution 1mAh)
- Electrical Rpm (resolution 100Rpm)

The theory

The ESC's „TLM“ or „TX“ pins can be wired togather to one „RX“ or halfduplex serial input pin of one serial connection of the receiving controller.
To need no ESC ID's the telemetry is requested with a short PWM pulse of 30µS (+-2µS). As the controller will have one signal trace to each ESC anyway.
This request can be done independent of the normal throttle signal. Also independent of the used signal speed. There is no limit (except of the 30µS duration) of the maximum requests per second. Every request will be answered.
The answer will be send with 115200 baud so one transmission will take ~900µS.

The transmission

One transmission will have 10 8-bit bytes sent with 115200 baud and 3.6V.

Byte 0: Temperature
Byte 1: Voltage high byte
Byte 2: Voltage low byte
Byte 3: Current high byte
Byte 4: Current low byte
Byte 5: Consumption high byte
Byte 6: Consumption low byte
Byte 7: Rpm high byte
Byte 8: Rpm low byte
Byte 9: 8-bit CRC

The CRC8

To validate the integrity of the received bytes you can check the CRC8.
This functions can be used to do it.

```
uint8_t update_crc8(uint8_t crc, uint8_t crc_seed){
        uint8_t crc_u, i;
        crc_u = crc;
        crc_u ^= crc_seed;
        for ( i=0; i<8; i++) crc_u = ( crc_u & 0x80 ) ? 0x7 ^ ( crc_u << 1 ) : ( crc_u << 1 );
        return (crc_u);
}

uint8_t get_crc8(uint8_t *Buf, uint8_t BufLen){
        uint8_t crc = 0, i;
        for( i=0; i<BufLen; i++) crc = update_crc8(Buf[i], crc);
        return (crc);
}
```

it dont needs much recources.

Converting the received values to standard units

int8_t Temperature = Temperature in 1°C
uint16_t Voltage = Volt *100 so 1000 are 10.00V
uint16_t Current = Ampere * 100 so 1000 are 10.00A
uint16_t Consumption = Consumption in 1mAh
uint16_t ERpm = Electrical Rpm /100 so 100 are 10000 Erpm

note: to get the real Rpm of the motor you will need to divide the Erpm result by the magnetpole count divided by two.
So with a 14magnetpole motor:
Rpm = Erpm/7

Wiring

We recommend to pull up the onewire signal one time with ~1k resistor to VCC (~2.5-5V - 3.3V typical) to avoid noise errors. (on the KISS FC this pullup is already onboard).