protocol.md

Table of contents

Nomenclature

- Node – A single device containing some number of sensors or/and actuators
- Gateway – Device which establishes and maintains the mesh network between nodes
- Sensor – Device, which converts some physical property to a voltage potential.
- Packet – A piece of data, which contains a series of measurements/commands to/from a node.

Limitations

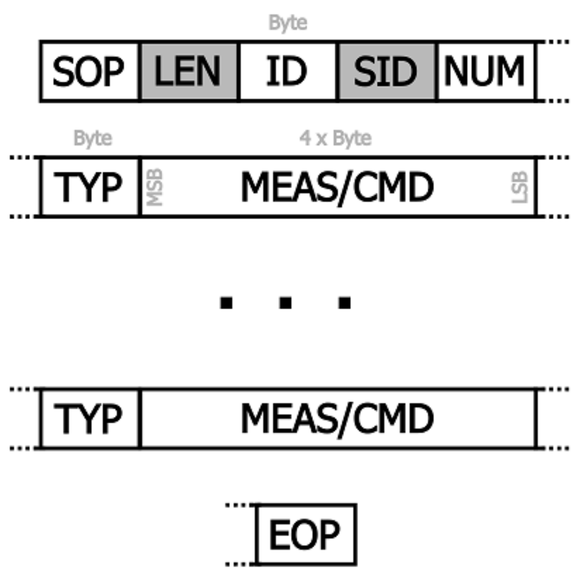
- Maximum number of measurements per packet is equal to 14.

Endianess

Big Endian, i.e, MSB byte always comes first

Packet Structure

The image below shows a build-up of a typical packet:



- SOP - Start of packet byte, always set to `0xFA`
- LEN - Number of packets in session, always set to `0x01`

- ID - Packet index, increments by one for each new packet
  - Index can not be zero, 1..255
- SID - Request id, always set to `0x00` , unless responding to a command
- NUM - Number of measurements/commands in this packet
- TYP - Measurement/Command type, see this table for valid types
- MEAS/CMD - Measurement/Command payload
- EOP - End of packet token, always set to `0xCE`

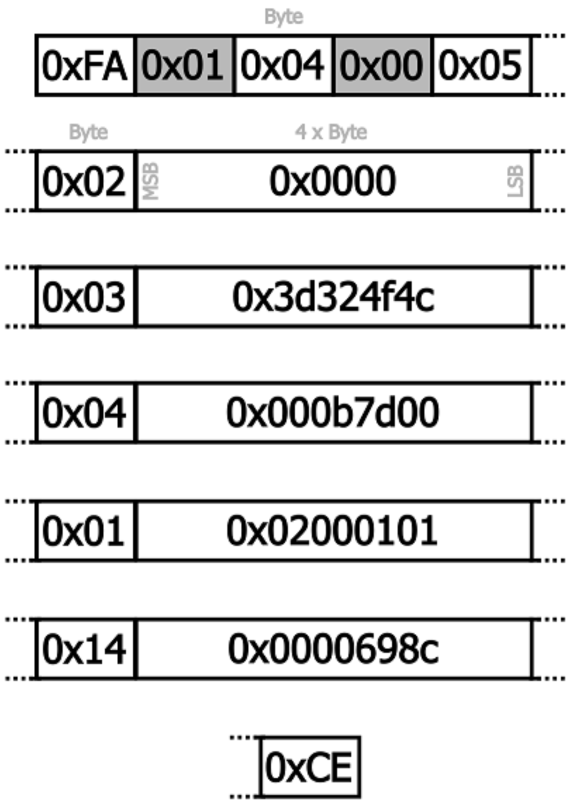
Measurement Types

Type	Id
No Measurement	0
Driver info	1
Sampling time MSB	2
Sampling time LSB	3
Sampling time Offset	4
On-die voltage	7
Battery voltage	8
On-die temperature	11
Voltage (real part)	13
Voltage (imaginary part)	14
Current (real part)	15
Current (imaginary part)	16
Charge	19
Temperature	20
Humidity	21
Pressure	22
Acceleration (x-axis)	23
Acceleration (y-axis)	24
Acceleration (z-axis)	25
Interrupt Event	26
Acoustic Level (average)	27
Acoustic Level (max)	28
Acoustic Level (dBSpl)	29
Ambient Light (visible)	30
Ambient Light (IR)	31
Ambient Light (UV)	32
CO2 Level	33
Distance	34
Sample Rate	35
Raw Value	123
SW Version	124

Type	Id
Driver Response	125
Packet Acknowledge	126
Error Code	127
CRC Code	128
Shutdown Event	129

Default Packet

Each packet delivered to gateway must contain the full 96 bit precision timestamp, the timestamp must be obtained from the `dn_getNetworkTime()` function. Each new measurement appended to the packet must be prepended by a `Driver Info` measurement unless the measurement comes from the same physical device, `Driver Info` helps us differentiate between measurements of the same type or helps us group measurements of different types sourced from a single physical device.



Consider the packet below:

It can be seen that the packet has id equal to 4, it contains 5 measurements according where the first three are time-stamp related, the next one tells us about the driver which has taken the measurement and the last one contains the actual measurement. From the type we can see that it is a temperature measurement. By decoding the packet we get the following:

```
+--+ id              : 004
+--+ request id      : 000
+--+ sample time     : 06:27:56 +0.75 sec
+--+ slot: 00, drv: 02, index: 01, ena: True
|  +-- temperature   : 25.60 C
+--+ eop
```

Each measurement type can have an individual conversion function associated with it in the cloud, so measurement type merely helps us identify the payload. For the example above, we have defined a conversion function for temperature as:

```
    def __get_external_temperature (self, measurement):
        temp = (measurement) * 175.72/65536 - 46.85
        return [temp,]
```