```
// Up to Date Apollon decoder
function Decoder(bytes, port) {

    // Conversion of signed integers
    function uncomplement(val, bitwidth) {
        var isnegative = val & (1 << (bitwidth - 1));
        var boundary   = (1 << bitwidth);
        var minval     = -boundary;
        var mask       = boundary - 1;
        return isnegative ? minval + (val & mask) : val;
    }

    var decoded = {};

    if (port === 1) {

        // Attributes
        decoded.base_id         = bytes[0] >> 4;
        decoded.major_version   = bytes[0] & 0x0F;
        decoded.minor_version   = bytes[1] >> 4;
        decoded.product_version = bytes[1] & 0x0F;

        // Telemetry
        decoded.up_cnt               = bytes[2];
        decoded.battery_voltage      = ((bytes[3] << 8) | bytes[4]) /
1000.0;
        decoded.internal_temperature = bytes[5] - 128;
        decoded.alarm                = bytes[6] ? "ALARM" : "NO ALARM";
        decoded.master_value         = ((bytes[7] << 8) | bytes[8]); // in
mm

        // Start of version specific fields
        var byte_cnt = 9;

        // ToF onboard
        if (bytes[1] & 0x01) {
            decoded.tof_status   = bytes[byte_cnt++];
            decoded.tof_distance = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]); // in mm
            decoded.tof_index    = bytes[byte_cnt++];
        }

        // Radar onboard
        if (bytes[1] & 0x02) {
            decoded.radar_status      = bytes[byte_cnt++];
            decoded.radar_no_peaks    = bytes[byte_cnt++];
            decoded.radar_distance_1  = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.radar_amplitude_1 = uncomplement((bytes[byte_cnt++] <<
8) | bytes[byte_cnt++], 16);
            decoded.radar_distance_2  = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.radar_amplitude_2 = uncomplement((bytes[byte_cnt++] <<
8) | bytes[byte_cnt++], 16);
            decoded.radar_distance_3  = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.radar_amplitude_3 = uncomplement((bytes[byte_cnt++] <<
8) | bytes[byte_cnt++], 16);
```

```
        }

        // ACC onboard
        if (bytes[1] & 0x04) {
            decoded.acc_status      = bytes[byte_cnt++] ? "Fehler" : "OK";
            decoded.acc_orientation = bytes[byte_cnt++];
            decoded.acc_open        = bytes[byte_cnt++];
            decoded.acc_open_cnt    = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
            decoded.acc_impact      = bytes[byte_cnt++] ? "Vandalismus" :
"OK";
        }

        // Hall onboard
        if (bytes[1] & 0x08) {
            decoded.hall_open     = bytes[byte_cnt++];
            decoded.hall_open_cnt = ((bytes[byte_cnt++] << 8) |
bytes[byte_cnt++]);
        }
    }
    return decoded;
}
```