Преобразование значения температуры из полученного массива с байтами.

Входящий буфер buff [] состоит из 8 байт:
buff [0]:0x01
buff [1] - buff [4]: значение температуры
buff [5], buff [6]: 0x00
buff [7]: контрольная сумма

Внутри функции, формирующей буфер, байты складываются следующим образом:

```
uint8_t* FillOutputBuffer(float temp_, uint8_t* buf_)
{
    memset(buf_, 0, sizeof(*buf_));
    buf_[0] = 0x01;
    union {
        float f;
        uint8_t b[4];
    } float_bytes;

    float_bytes.f = temp_;

    buf_[1] = float_bytes.b[0];
    buf_[2] = float_bytes.b[1];
    buf_[3] = float_bytes.b[2];
    buf_[4] = float_bytes.b[3];
    buf_[5] = 0x00;
    buf_[6] = 0x00;

    uint8_t checksum = 0;
    for (int i = 0; i < 7; i++) {
        checksum ^= buf_[i];
    }
    buf_[7] = checksum;

    return buf_;
}
```

Контрольная сумма рассчитывается через XOR следующим образом:
checksum=buf[0]⊕buf[1]⊕buf[2]⊕...⊕buf[6]
И результат записывается в buff [7].

Полученный массив выглядит как показано ниже:

	buff [0]	buff [1]	buff [2]	buff [3]	buff [4]	buff [5]	buff [6]	buff [7]
DEC	1	154	153	181	65	0	0	246
HEX	0x01	0x9A	0x99	0xB5	0x41	0	0	0xF6
BIN	0001	100011010	10011001	10110101	01000001	0	0	11110110

Получение значений:

1. Проверка контрольной суммы.
   Ниже показан пример функции проверки контрольной суммы (на языке Си):

```
int check_checksum(uint8_t *buf) {
    uint8_t xor_sum = 0;
    for (int i = 0; i < 8; i++) {
        xor_sum ^= buf[i];
    }
    return (xor_sum == 0); // 1 - верная, 0 - ошибка
}
```

$0x01_{16} \oplus 0x9A_{16} = 0x9B_{16}$

$0x9B_{16} \oplus 0x99_{16} = 0x02_{16}$

$0x02_{16} \oplus 0xB5_{16} = 0xB7_{16}$

$0xB7_{16} \oplus 0x41_{16} = 0xF6_{16} = 246_{10}$

5-й и 6-й байт равны 0, поэтому контрольная сумма, записываемая в 7-й байт равна $0xF6_{16}$.

2. Получение значения температуры.

Ниже показан пример функции получения значения температуры (на языке Си):

```
float extract_float(uint8_t *buf_) {
    union {
        float f;
        uint8_t b[4];
    } float_bytes;

    float_bytes.b[0] = buf_[1]; // младший байт
    float_bytes.b[1] = buf_[2];
    float_bytes.b[2] = buf_[3];
    float_bytes.b[3] = buf_[4]; // старший байт

    return float_bytes.f;
}
```

В примере выше пришли данные $0x41\ B5\ 99\ 9A_{16} = 01000001\ 10110101\ 10011001\ 10011010_2$
При разборе этого значения по байтам получается, что 31-й байт показывает знак. 0 – положительное число. $10000011_2 = 131_{10}$ порядок (8 бит)
Реальный порядок = 131 - 127 = 4
Мантисса (23 бита): 01101011001100110011010
Мантисса хранится без ведущей единицы, поэтому $1.01101011001100110011010_2$
$1.01101011001100110011010_2 \approx 1.41875005_{10}$
$1 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3} + 0*2^{-4} + 1*2^{-5} + 0*2^{-6} + 1*2^{-7} + 1*2^{-8} + \ldots$

Мантисса: `1.01101011001100110011010`

Бит	Значение	Десятичное	Сумма
1 (целая часть)	1	1	1
0	2^{-1}	0	1
1	2^{-2}	0.25	1.25
1	2^{-3}	0.125	1.375
0	2^{-4}	0	1.375
1	2^{-5}	0.03125	1.40625
0	2^{-6}	0	1.40625
1	2^{-7}	0.0078125	1.4140625
1	2^{-8}	0.00390625	1.41796875
0	2^{-9}	0	1.41796875
0	2^{-10}	0	1.41796875
1	2^{-11}	0.00048828125	1.41845703125
1	2^{-12}	0.000244140625	1.418701171875
0	2^{-13}	0	1.418701171875
0	2^{-14}	0	1.418701171875
1	2^{-15}	0.000030517578125	1.418731689453125
1	2^{-16}	0.0000152587890625	1.4187469482421875
0	2^{-17}	0	1.4187469482421875
0	2^{-18}	0	1.4187469482421875
1	2^{-19}	0.0000019073486328125	1.4187488555908203
1	2^{-20}	0.00000095367431640625	1.4187498092651367
0	2^{-21}	0	1.4187498092651367
1	2^{-22}	0.000000238418579101562 5	1.4187500476837158

Расчёт значения температуры:

Значение температуры = $(-1)^{\text{знак}} \times 1.\text{мантисса} \times 2^{\text{порядок-127}}$

Значение температуры = $1{,}41875005 \times 2^4 = 22{,}7000008$