Протокол обмена с сервером УСПД ExpDevice LRW

1. Введение

Конечное устройство ExpDevice LRW передаёт показания счётчиков импульсов по сети LoRaWAN. В документе приведены сведения, необходимые для регистрации узла в LoRaWAN-сервере (например, ChirpStack), создания парсера uplink-пакетов и формирования корректных даунлинков.

Таблица 1 – основные параметры радиоинтерфейса

Параметр	**Значение**
Модуляция	LoRa (SX1262)
Мощность передатчика	до 16 дБм (40 мВт)
Чувствительность приёмника	до −137 дБм (SF12/BW125)
Частотный план по умолчанию	KZ865 (см. раздел 2)
Класс устройства	Class A
Способ регистрации	OTAA и ABP (по умолчанию ABP)
Тип uplink-сообщений	Unconfirmed (переключается командой)
Период отправки в сеть	5 мин, 15 мин, 30 мин, 1 ч, 6 ч, 12 ч, 24 ч; значение по умолчанию — 12 ч

2. Частотный план

Устройство использует каналы, предусмотренные регионом KZ865 (LoRaWAN Regional Parameters 1.0.3). По умолчанию активны три uplink-канала и одна частота RX2. Дополнительные каналы (865.7–865.9 МГц) допускается добавлять через конфигуратор.

Таблица 2 – параметры основных каналов

Канал	**Частота, МГц**	**Spreading Factor**	**Полоса, кГц**
1	865.1	SF12–SF7	125
2	865.3	SF12–SF7	125
3	865.5	SF12–SF7	125
RX2	866.7	SF12	125

3. Сбор и передача данных

1. Импульсы с дискретных входов (до 16 каналов) аккумулируются во внутренних 32-битных регистрах и периодически сохраняются во встроенной энергонезависимой памяти.
2. Перед Formированием uplink-пакета устройство измеряет температуру окружающей среды, оценивает относительный уровень батареи и фиксирует временную метку Unix Epoch (UTC).

3. Периодичность передачи выбирается из набора 5 мин, 15 мин, 30 мин, 1 ч, 6 ч, 12 ч и 24 ч (значение по умолчанию — 12 ч) и может изменяться как локально, так и по сети (раздел 4.2).
4. При включении подтверждённого режима устройство следует стандартному механизму повторов LoRaWAN до получения подтверждения.

4. Описание пакетов данных

Все числовые поля передаются в формате little-endian (младший байт первым).

4.1 Периодические показания (порт 2)

Uplink-пакет телеметрии имеет тип 0x02. ExpDevice LRW варианты от 2 до 10 каналов, но у них у всех одна структура пакета.

Таблица 3 – структура uplink-пакета

Смещение	Размер, байт	Описание
0	1	Тип пакета 0x02.
1	1	Температура, °C (int8_t, младший байт).
2	1	Заряд батареи, %: (raw × 100) / 254, где raw — уровень батареи LoRaWAN (1…254).
3	4	Время измерения в Unix Epoch (UTC).
7	1	Маска активных каналов 0–7 (таблица 4).
8	1	Маска активных каналов 8–15.
9	N	Закодированные значения счётчиков всех активных каналов (таблица 5).

Маски позволяют определить, какие счётчики присутствуют в потоке.

Таблица 4 – разметка битов маски активных каналов

Бит	Канал	Описание
0–7	0–7	Если бит установлен, значение соответствующего канала присутствует в полезной нагрузке.
8–15	8–15	Второй байт маски: бит = номер канала − 8.

Если устройство сконфигурировано на большее количество каналов, в одном пакете участвуют только младшие 16, поскольку маска состоит из двух байт.

Первые два бита последнего байта каждого значения кодируют длину последовательности: 00 — один байт, 01 — два, 10 — три, 11 — четыре. Остальные биты содержат старшие разряды значения.

Таблица 5 – переменное кодирование значений счётчиков

Диапазон значения счётчика	**Количество байт**	**Формат**	**Комментарий**
0x000000–0x00003F	1	[00]xxxxxx	Один байт, 6 младших бит несут значение.
0x000040–0x003FFF	2	LSB, [01]xxxxxx	Второй байт содержит биты [13:8] и префикс 01 (0x40).
0x00004000–0x3FFFFF	3	LSB, Mid, [10]xxxxxx	Третий байт несёт биты [21:16] и префикс 10 (0x80).
0x00400000–0x3FFFFFFF	4	LSB, Mid0, Mid1, [11]xxxxxx	Четвёртый байт содержит биты [29:24] и префикс 11 (0xC0).

Устройство проходит по каналам по возрастанию; при ненулевом значении сначала устанавливает бит маски, затем добавляет значение в поток.

Пример полезной нагрузки для каналов 0 и 1 (25 и 0x34567), заряда 100 % и времени 0x64B3A5C0:

02 | 19 | 64 | C0 A5 B3 64 | 03 00 | 19 | 67 45 83

4.2 Настройка параметров (порт 3)

Даунлинк-пакеты на порт 3 меняют ключевые настройки. Payload[0] определяет тип команды:

- Payload[0] = 0 — список параметров: начиная со второго байта передаётся последовательность пар [ParamId][Value] по одному байту.
- Payload[0] = 1 — зарезервировано под переключение класса (в текущей версии команды отсутствуют).

После применения валидных команд настройки сохраняются во внутренней Flash-памяти и не теряются при перезапуске.

Таблица 6 – допустимые параметры конфигурации

ParamId	**Value**	**Описание**
0	0 – unconfirmed, 1 – confirmed	Переключение типа uplink-сообщений.
1	Любой	Зарезервировано под количество повторов (не реализовано).

2	Любой	Зарезервировано под режим портов (не реализовано).
3	0…6	Выбор периода передачи (таблица 7).

Таблица 7 – соответствие кода периода и тайм-аута

Значение	**Период**
0	5 минут
1	15 минут
2	30 минут
3	1 час
4	6 часов
5	12 часов
6	24 часа

5. Сохранение и восстановление параметров

- Устройство периодически сохраняет накопленные показания счётчиков во внутреннюю энергонезависимую память, поэтому значения не теряются при отключении питания.
- Пользовательские настройки (тип передачи, периодичность, способ активации OTAA/ABP) также хранятся во Flash и автоматически восстанавливаются после перезапуска.

6. Пример парсера

```
def parse_data(data_bytes):
    TEMP_INDEX = 1
    BAT_LEVEL_INDEX = 2
    TIMESTAMP_START_INDEX = 3
    TIMESTAMP_END_INDEX = 7
    PORTS_STATE_FLAG_START_INDEX = 7
    PORTS_STATE_FLAG_END_INDEX = 9
    MAX_PORTS = 10


    temp = data_bytes[TEMP_INDEX]
    bat_level = data_bytes[BAT_LEVEL_INDEX]
    timestamp = int.from_bytes(
```

```
    data_bytes[TIMESTAMP_START_INDEX:TIMESTAMP_END_INDEX],
    byteorder="little"
)
ports_state_flag = int.from_bytes(
    data_bytes[PORTS_STATE_FLAG_START_INDEX:PORTS_STATE_FLAG_END_INDEX],
    byteorder="little"
)
ports_state = {
    port: bool(ports_state_flag & (1 << port))
    for port in range(MAX_PORTS - 1, -1, -1)
}
imps = [0] * MAX_PORTS
reversed_data = data_bytes[::-1]
data_array = bytearray(reversed_data)
index = 0
for port, state in ports_state.items():
    if state:
        num_of_bytes = (data_array[index] >> 6) + 1
        data_array[index] &= 0x3F
        port_value = int.from_bytes(
            data_array[index: index + num_of_bytes],
            byteorder='big'
        )
        imps[port] = port_value
        index += num_of_bytes
return temp, bat_level, timestamp, imps
```