1. Протокол уровня приложения SMP-M. Бинарный позиционный протокол данных

2. Модель отправки/получения пакетов прибором

3. [Счетчики Тепла] Пакет дневных показаний для счетчиков тепла (16байт). Uplink

4. [Счетчики Воды] Пакет дневных показаний для счетчиков воды с маской событий и меткой времени (16байт). Uplink

5. [Счетчики Воды] Пакет дневных показаний для счетчиков воды с запорным клапаном и маской событий (8 байт). Uplink

6. [Счетчики Импульсов] Пакет текущих показаний объёма для счетчиков импульсов (12байт). Uplink

7. [Счетчики электрической энергии. Общий] Пакет ответа на Downlink запрос. Uplink

8. [Счетчики электрической энергии] Пакет показаний энергии с ретроспективой показаний и за последние 3 дня/месяца, общее или по тарифу. Uplink

9. [Счетчики электрической энергии] Пакет событий. Uplink

10. [Счетчики электрической энергии] Пакет показаний потребленной электроэнергии по 3м фазам. Uplink

11. [Счетчики электрической энергии] Пакет показаний сгенерированной электроэнергии по 3м фазам. Uplink

12. [Счетчики электрической энергии] Пакет общих показаний для счетчиков электроэнергии с маской событий. Uplink

13. [Счетчики электрической энергии] Информационный пакет с параметрами электросети счетчика электроэнергии. Uplink

14. [Счетчики электрической энергии] Информационный пакет счетчика электроэнергии. Uplink

15. [Счетчики электрической энергии] Пакет дневных показаний профилей показаний электросчетчика по часам (00:00-08:00). Uplink

16. [Счетчики электрической энергии] Пакет дневных показаний профилей показаний электросчетчика по часам (08:00-16:00). Uplink

17. [Счетчики электрической энергии] Пакет дневных показаний профилей показаний электросчетчика по часам (16:00-24:00). Uplink

18. [Счетчики электрической энергии] Пакет показаний тарифов потребления электроэнергии. Uplink

19. [Счетчики электрической энергии] Пакет показаний тарифов сгенерированной электроэнергии. Uplink

20. [Счетчики электрической энергии] Пакет установки регулярной отправки пакетов с данными. Downlink

21. [Счетчики электрической энергии] Пакет запроса пакетов данных (16байт). Downlink

22. [Счетчики электрической энергии] Пакет запроса пакетов данных укороченный (8байт). Downlink

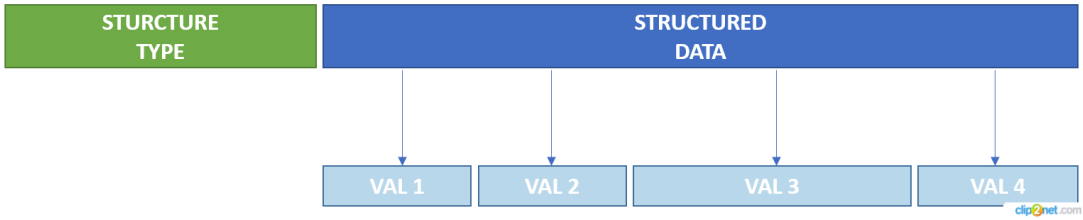
23. [Счетчики электрической энергии] Установка Времени (8байт). Downlink

24. [Счетчики электрической энергии] Установка состояния реле. Downlink

Протокол уровня приложения SMP-M. Бинарный позиционный протокол данных

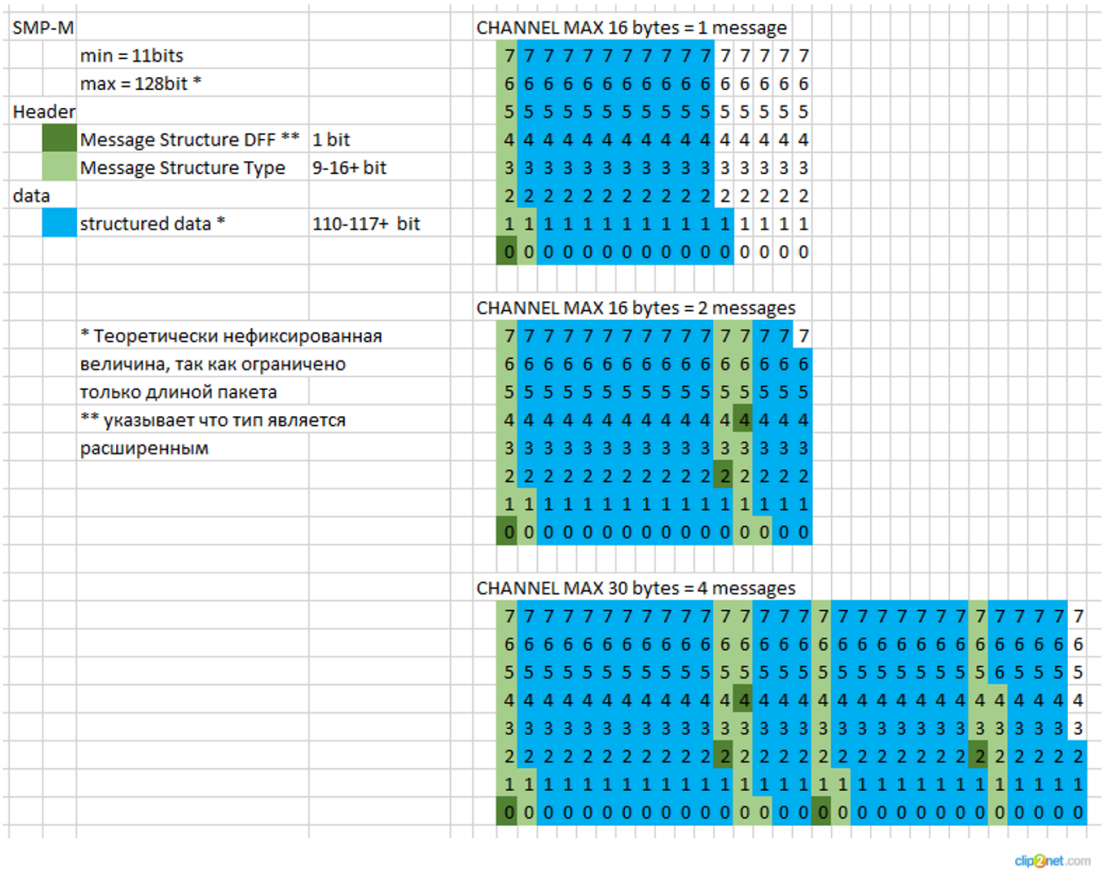
Структура пакета SMP-M

Однозначное представление данных по ID Пакета (по типу структуры)



Протокол разрабатывался с учетом возможности занять любой канал и все доступное ему место (начиная с минимально доступного - 4байт)

Для того что бы занять любое доступное место, сообщения могут идти в одном пакете друг за другом. Оставшееся пространство предлагается заполнить нулями



Если пакет не может поместиться в рамки размера канала, поэтому протокол более низкого уровня (например транспортный или канальный) должен обеспечить возможность передачи сообщения, разбивая сообщение на фреймы. Если такой возможности у протокола более низкого уровня нет - сообщение не может быть передано

Описание

- Строгая структура протокола
  - Однозначность парсинга и сериализации данных
  - Значения всегда стоят на своих местах и подчиняются только своим правилам
- Каждое сообщение самодостаточно

    - Однозначность и самодостаточность данных в пакете. Один пакет может быть использован как всвязке с остальными пакетами, так и отдельно. Тип пакета указывает строго на те данные которые придут
    - Для этого нужно достаточное место для данных
    - Для регулярных сообщений можно вариьировать точностью и наполнением пакета
- Возможность занять любой канал (отведенный размер полезных данных)
    - Пакеты SMPM выстраиваются друг за другом, Остаток фиксированного канала заполняется нулями
    - Если парсер, разбирая полезные данные последовательно не может определить что это за пакет, то весь канальный пакет (с любым количеством SMPM пакетов на борту) отбрасывается
    - Это развязывает руки для широких каналов, увеличивает применяемость протокола, не меняя реализацию пакета данных на стороне прибора. эффективней тратятся ресурсы прибора
    - Отправка нескольких SMPM пакетов через один канальный пакет приводит к минимальному размеру заголовков, на единицу полезных данных
    - Минимальный канал получает минимальное наполнение данными
- Возможность расширить ID пакетов.
    - Использование Dynamic Field Flag (DFF) для указания типа сообщения с минимальными накладными расходами на место в пакете. Блокировка расширения типа должна быть как можно позже = выбор сложного шага DFF
    - Пакеты SMPM имеют гибкую структуру ID. Сегменты DFF могут быть расширены до бесконечности, закрыв все возможные кейсы в будущем
    - Это полезно при необходимости расширять протокол, не разрабатывая новых протоколов, дополняя существующее решение

Сквозное пространство идентификаторов пакетов.

Принципы при проектировании пакета:

- ID Пакета это число. от 0 до бесконечности = Числовое однозначное представление о составе и типе пакета
- Каждый пакет должен иметь понятную предсказуемую структуру, которая и определена самим ID Пакета.
- Для UpLink и для DownLink имеют различные пространства чисел для ID Пакетов
- Сам тип/структуры данных ID пакета должен иметь возможность отображать эффективное хранение для разных типов каналов
- Последующие версии протокола не имеют права заменять ID пакета, которое было определено в версии протокола ниже

Dynamic Field Flag (DFF)

Так как это статический бинарный протокол, то есть необходимость расширять пакеты, имея простой механизм.

Таким механизмом стал DFF.

Алгоритмически число делится на сегменты. Каждый сигмент имеет DFF, который говорит есть ли за ним следующий сегмент, или нет.

Сегменты в конкретных случаях могут быть различной величины.

Для `package_type_id` для протокола SMPM выбран сегменты 8-3-3.

Это означает, что первый сегмент всегда занимает 8бит (один бит из них это DFF), а все последующие имеют длину 3 (один бит из них это DFF).

#	Длина сегмента (бит)	Первое число в сегменте	Последнее число в сегменте	Сумма накладных расходов (бит)
1	8	1	127	1
2	3	128	511	2
3	3	512	2047	3
4	3	2048	8191	4
...	3	...	....	...

Это позволило разбить на сегменты идентификаторы, и для пакеты малого размера тратить на заголовок минимальный размер, при этом не жертвуя сильно пространством.

Меньше чем 8 не имеет смысла, так как пространство чисел сократится минимум вдвое, и то же количество идентификаторов придется записывать сбольшими накладными расходами.

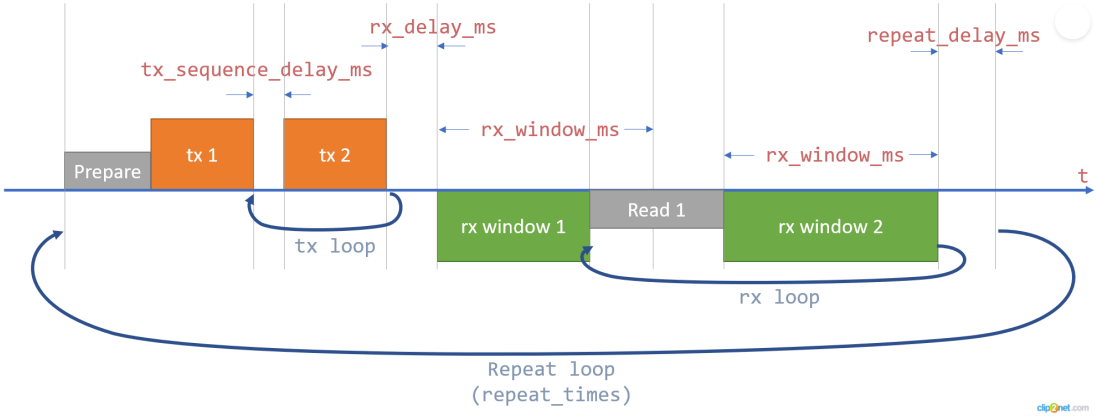
Так же это позволяет иметь пакеты ровно в размер заголовка = 1байт (8бит).

Последнее обновление: 2023-06-29

Модель отправки/получения пакетов прибором

> **Важное требование, призванное упростить логику работы всей системы**
>
> И приборы с постоянным питанием и приборы с батарейным питанием работают одинаково !

Цикл Rx/Tx



Основные настройки цикла:

- `schedule` - расписание по которому выходит на связь и отправляет данные конкретного типа `message_type`
- `message_type` - тип данных которые нужно отправить по расписанию `schedule`
- `repeat_times` - количество повторений сообщений
- `repeat_delay_ms` - задержка во времени между повторами сообщений
- `sending_delay_ms` - задержка во времени между отправлением сообщений
- `sleep_after_send_ms` - задержка во времени перед тем как слушать эфир на прием
- `rx_window_ms` - временное окно, за которое прибор пытается получить от БС команду

Описание цикла Rx/Tx

```
0: Просыпается в заданное ему время и расписание `schedule`,
   которое может содержать конкретную дату, день недели или конкретное время выхода на связь
1: Определят какие типы сообщений нужно отправить сейчас
2: Если включен режим `ListenBeforeTalk`
    2.0: Начинает прослушивание эфира в поисках момента, когда можно начинать отправку
3: В цикле повтора сообщений, количество раз заданно в `repeat_times`
    3.0: в цикле по всем нужным сообщениям, которые нужно отправить
        3.0.0: идет за данными в хранилище и получает нужные значения
               для сообщения, что нужно отправить сейчас
        3.0.1: формирует сообщение и сериализует в формат необходимого протокола,
               на выходе получает буфер бит, которые нужно отправить в эфир
        3.0.2: отправляет данные по средством UPLINK радио канала к БС
        3.0.3: если есть еще сообщения, которые нужно отправить то заспыет на `sending_delay_ms`
    3.1: засыпает на время `sleep_after_send_ms`
    3.2: Сбрасывает буфер полученных сообщений, в цикле ожидает управляющих сообщений от БС
        3.2.0: просыпается и ожидает `rx_window_ms` получения нужной преамбулы
               ожидаемого им протокола DOWNLINK от БС.
               Фильтрует сообщения по маку фильтрации
               (то что это сообщение именно ему адресовано или бродкаст)
        3.2.1: если за это время получил валидное для него сообщение,
               то читает сообщение до конца (установлено протоколом DOWNLINK),
               складывает сообщение в буфер и засыпает на `rx_delay_ms`
    3.3: Обрабатывает полученный буфер сообщений
    3.4: определяет надо ли отправлять какие либо сообщения в ответ Базовой станции
    3.5: если сообщений отправлять не надо и ему ничего не пришло от БС,
         в случае если количество `repeat_times` не истекло, то
        3.5.0: засыпает на время `repeat_delay_ms`
        3.5.1: цикл повторяется с пункта 3.0
4: Если необходимо (для счетчиков с постоянным энергопитанием
   это может быть включено по умолчанию),
   то ожидает приема сообщения в режиме постоянного прослушивания эфира
5: засыпает до следующего сеанса
```

Последнее обновление: 2023-06-29

[Счетчики Тепла] Пакет дневных показаний для счетчиков тепла (16байт)

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_heat_proxy_meter_16b_daily`

Описание

Пакет показаний теплосчетчика, собираемый прибором радио-модулем-накладкой от прибора учета тепла.

Пакет отправляется раз в день.

Время выхода на связь и количество раз (отправляемых пакета в эфир этот) регламентируется настройкой прибора.

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	17 бит	const(dff(8,3,3,3...))	-	-	-	25732 (*bin* **0 01100100 10000100**)
UNUSED	-	15 бит	-	0	0	-	-
value	значение потребления	27 бит	uf27p3	0.0	-	134217.727	-
UNUSED	-	5 бит	-	0	0	-	-
uptime_min	количество целых минут без ошибок	22 бит	u22	0	-	4194303	-
meter_battery_volts	уровень заряда батареи (в вольтах)	9 бит	uf9p2	0.0	5.11	-	-
UNUSED	-	1 бит	-	0	0	-	-
capacitor_volts	уровень заряда батареи (в вольтах)	9 бит	uf9p2	0.0	5.11	-	-
radio_proxy_battery_volts	уровень заряда батареи радиомодуля (в вольтах)	9 бит	uf9p2	0.0	5.11	-	-
error_meter_sync	флаг указания на проблему с синхронизацией со счетчиком	1 бит	bool	0	1	-	-
error_reset	флаг указания на сброс радио модуля	1 бит	bool	0	1	-	-
RESERVED	-	12 бит	-	-	-	-	

Референсные значения типов данных

value

Значение потребленного (отпущенного) количества теплоты (тепловой энергии)

значение потребления

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `3`
- Минимальное значение: `0.0`

- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 134217.727`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `27 бит`
- тип данных: `uf27p3`

uptime_min

Количество целых минут без ошибок (с момента последней)

количество целых минут без ошибок

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 4194303`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `22 бит`
- тип данных: `u22`

meter_battery_volts

Уровень заряда батареи (в вольтах). мгновенное значение в вольтах заряда батареи, запрашиваемое у счетчика

уровень заряда батареи (в вольтах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `5.11`
- Лимитирование значения: `Значение принимает 0.0 если меньше и 5.11 если больше`
- Переполнение значения: `Не допустимо`
- Размер: `9 бит`
- тип данных: `uf9p2`

capacitor_volts

Уровень заряда батареи (в вольтах). перед отправкой пакета устройство должно измерить значение напряжения на своем источнике питания и ионисторе

уровень заряда батареи (в вольтах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`

- Минимальное значение: `0.0`
- Максимальное значение: `5.11`
- Лимитирование значения: `Значение принимает 0.0 если меньше и 5.11 если больше`
- Переполнение значения: `Не допустимо`
- Размер: `9 бит`
- тип данных: `uf9p2`

radio_proxy_battery_volts

Уровень заряда батареи (в вольтах). должно замеряться перед отправкой пакета вместе со значением ионистора

уровень заряда батареи радиомодуля (в вольтах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `5.11`
- Лимитирование значения: `Значение принимает 0.0 если меньше и 5.11 если больше`
- Переполнение значения: `Не допустимо`
- Размер: `9 бит`
- тип данных: `uf9p2`

error_meter_sync

флаг указания на проблему с синхронизацией со счетчиком

- Размер: `1 бит`
- тип данных: `bool`

error_reset

флаг указания на сброс радио модуля

- Размер: `1 бит`
- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDevWaterPackId.UL_DATA_16B__HEAT_PROXY_DAILY(2052)
value: 3.3
uptime_min: 3
meter_battery_volts: 3.3
capacitor_volts: 3.33
radio_proxy_battery_volts: 3.31
```

```
error_meter_sync: false
error_reset: false
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	4	0000100
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	0	00
packet_type_id.1.DFF	bool	1	1	1
packet_type_id.2.VAL	u2	2	0	00
packet_type_id.2.DFF	bool	1	1	1
packet_type_id.3.VAL	u2	2	1	01
packet_type_id.3.DFF	bool	1	0	0
UNUSED	-	15	0	000000000000000
value	uf27p3	27	3300	000000000000000110011100100
UNUSED	-	5	0	00000
uptime_min	u22	22	3	0000000000000000000011
meter_battery_volts	uf9p2	9	330	101001010
UNUSED	-	1	0	0
capacitor_volts	uf9p2	9	333	101001101
radio_proxy_battery_volts	uf9p2	9	331	101001011
error_meter_sync	bool	1	0	0
error_reset	bool	1	0	0
RESERVED	u12	12	0	000000000000

Result BIN (MSB - LSB):

```
00000000 00000010 10010111 01001101 01010010 10000000 00000000 00000011 00000000 00000000 00001100 11100100 00000000 00000000 01100100 10000100
```

Result HEX (little-endian):

```
84 64 00 00 e4 0c 00 00 03 00 80 52 4d 97 02 00
```

Парсинг/Сериализация значений

Python C

```python
import math
import typing


# PACKET (128 bits)   smpm_ul_device_heat_proxy_meter_16b_daily
#
# RESULT int:        1345336366824802144628629480199898O
# RESULT bin:  MSB   00000000 00000010 10010111 01001101 01010010 10000000 00000000 00000011 00000000 00000000 00001100 11100100 00000000 00000000 01100100 10000100   LSB
# RESULT hex:  LE    84 64 00 00 e4 0c 00 00 03 00 80 52 4d 97 02 00
```

```
#
# name                      type    size  value(int)                                                                                                                          data(bits)
# ------------------------- ------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL      u7         7           4                                                                                                                             0000100
# packet_type_id.0.DFF      bool       1           1                                                                                                                            1
# packet_type_id.1.VAL      u2         2           0                                                                                                                          00
# packet_type_id.1.DFF      bool       1           1                                                                                                                         1
# packet_type_id.2.VAL      u2         2           0                                                                                                                       00
# packet_type_id.2.DFF      bool       1           1                                                                                                                      1
# packet_type_id.3.VAL      u2         2           1                                                                                                                    01
# packet_type_id.3.DFF      bool       1           0                                                                                                                   0
# UNUSED                    -         15           0                                                                                                    000000000000000
# value                     uf27p3    27        3300                                                                       000000000000000110011100100
# UNUSED                    -          5           0                                                                  00000
# uptime_min                u22       22           3                                            0000000000000000000011
# meter_battery_volts       uf9p2      9         330                                   101001010
# UNUSED                    -          1           0                                  0
# capacitor_volts           uf9p2      9         333                         101001101
# radio_proxy_battery_volts uf9p2      9         331                101001011
# error_meter_sync          bool       1           0               0
# error_reset               bool       1           0              0
# RESERVED                  u12       12           0  000000000000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceHeatProxyMeter16BDailyData(typing.NamedTuple):
    value: float
    uptime_min: int
    meter_battery_volts: float
    capacitor_volts: float
    radio_proxy_battery_volts: float
    error_meter_sync: bool
    error_reset: bool

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((4) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((0) & (2 ** (2) - 1)) << size
```

```python
        size += 2
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((0) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((0) & (2 ** (15) - 1)) << size
        size += 15
        assert isinstance(data.value, (int, float))
        result |= ((int(round(float(data.value) * 1000.0, 0)) & 134217727) & (2 ** (27) - 1)) << size
        size += 27
        result |= ((0) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.uptime_min, int)
        result |= (((data.uptime_min) & 4194303) & (2 ** (22) - 1)) << size
        size += 22
        assert isinstance(data.meter_battery_volts, (int, float))
        result |= ((int(round(max(min(5.11, float(data.meter_battery_volts)), 0.0) * 100.0, 0))) & (2 ** (9) - 1)) << size
        size += 9
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.capacitor_volts, (int, float))
        result |= ((int(round(max(min(5.11, float(data.capacitor_volts)), 0.0) * 100.0, 0))) & (2 ** (9) - 1)) << size
        size += 9
        assert isinstance(data.radio_proxy_battery_volts, (int, float))
        result |= ((int(round(max(min(5.11, float(data.radio_proxy_battery_volts)), 0.0) * 100.0, 0))) & (2 ** (9) - 1)) << size
        size += 9
        assert isinstance(data.error_meter_sync, bool)
        result |= ((int(data.error_meter_sync)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_reset, bool)
        result |= ((int(data.error_reset)) & (2 ** (1) - 1)) << size
        size += 1
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceHeatProxyMeter16BDailyData':
        result__el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 4 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        buf.shift(15)
        result__el_tmp1["value"] = round(buf.shift(27) / 1000.0, 3)
        buf.shift(5)
        result__el_tmp1["uptime_min"] = buf.shift(22) + 0
        result__el_tmp1["meter_battery_volts"] = round(buf.shift(9) / 100.0, 2)
        buf.shift(1)
        result__el_tmp1["capacitor_volts"] = round(buf.shift(9) / 100.0, 2)
        result__el_tmp1["radio_proxy_battery_volts"] = round(buf.shift(9) / 100.0, 2)
```

```
        result__el_tmp1["error_meter_sync"] = bool(buf.shift(1))
        result__el_tmp1["error_reset"] = bool(buf.shift(1))
        result = SmpmUlDeviceHeatProxyMeter16BDailyData(**result__el_tmp1)
        buf.shift(12)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Воды] Пакет дневных показаний для счетчиков воды с маской событий и меткой времени (16байт)

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_water_meter_16b_daily`

Описание

Пакет отправляется каждый день дважды (на разных каналах). Первый раз - в промежутке с 00:00-12:00. Второй с 12:00-00:00

Наличие конкретного события (за период времени от последнего дневого пакета) указывается одним единичным битом

Расчет значений (берутся из дневных профилей потребления) за предыдущие дни (например сейчас 2022-12-16 20:13:03)

1а. берется точное значение (параметр direct_flow_volume) и отнимается значение на момент закрытия предыдущего дня (2022-12-16 00:00:00);

1б. direct_flow_volume_day_ago = полученное число округляется математически (0.5 -> 1);

2а. берется точное значение (параметр direct_flow_volume) и отнимается значение на момент закрытия 2 дня назад (2022-12-15 00:00:00);

2б. от полученного числа отнимается direct_flow_volume_day_ago (без округления);

Отрицательным значение direct_flow_volume_day_ago быть не может. Число 0.0 должно игнорироваться на верхнем уровне. Максимальное число 12.7 считается ошибкой и не берется в расчеты на стороне верхнего уровня.

В случае если нет синхронизации времени значения берутся из момента отправки за конкретные дни (на момент второго сообщения за день). Поэтому прибор должен хранить точниые значения двух последних дней (ориентировочно 24ч между собой)

Пакет может быть отправлен только в рамках температурного режима от +5 градусов Цельсия и выше

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	14 бит	const(dff(8,3,3,3...))	-	-	-	3203 (*bin* **001100 10000011**)
days_ago	показание на конкретные сутки (дней назад)	5 бит	timedelta	0 секунд	31 день	-	-
sync_time_days_ago	количество целых дней с последней синхронизации	3 бит	timedelta	0 секунд	7 дней	-	-
timestamp_s	Время в секундах от 2020-01-01 00:00:00.	26 бит	timedelta	0 секунд	-	776 дней 17 часов 21 минута 3 секунды	-
temperature	температура (с точностью до 1 градуса цельсия)	7 бит	u7	-35	92	-	-
battery_volts	уровень заряда батареи (в вольтах)	6 бит	uf6p1	0.0	6.3	-	-
event_reset	Была перезагрузка МК по какой то причине	1 бит	bool	0	1	-	-
event_low_battery_level	Указание что батарейка разряжена	1 бит	bool	0	1	-	-
event_temperature_limits	Выход за рамки эксплуатационных норм по температуре	1 бит	bool	0	1	-	-
direct_flow_volume	объем прямого потока (в м3)	32 бит	uf32p3	0.0	-	4294967.295	-
direct_flow_volume_day_ago	разница объема прямого потока (в м3) 1 день назад	7 бит	uf7p1	0.0	12.7	-	-
reverse_flow_volume	объем обратного потока (в м3)	12 бит	uf12p2	0.0	-	40.95	-
event_battery_warn	Предупреждение о низком заряде (1й порог) батареи	1 бит	bool	0	1	-	-
event_system_error	Указание что была внутренняя ошибка ПО. программа сама обработала сбой	1 бит	bool	0	1	-	-

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
event_flow_reverse	Событие превышения лимита обратного хода	1 бит	bool	0	1	-	-
event_flow_speed_is_over_limit	Указание что было превышение лимитов скорости поотока	1 бит	bool	0	1	-	-
event_sensor_error	Событие неисправности сенсора потока (например УЗ датчика)	1 бит	bool	0	1	-	-
event_sensor_error_temperature	Указание что был сбой внешнего датчика температуры	1 бит	bool	0	1	-	-
event_case_was_opened	Указание что было отрытие корпуса	1 бит	bool	0	1	-	-
event_continuous_consumption	Событие непрерывного потребления (потенциальная утечка)	1 бит	bool	0	1	-	-
event_no_resource	Событие нет ресурса (газа)	1 бит	bool	0	1	-	-
event_magnet	Указание что магнит присутствует на момент отправки	1 бит	bool	0	1	-	-
RESERVED	-	3 бит	-	-	-	-	

Референсные значения типов данных

days_ago

Количество дней назад. 0 - текущие показания на момент отправки сообщения.
Например 3 означает на конец 3 дня от текущего. отсчет дня и времени идет в той тайм зоне которой работает прибор (сегодня 2022-11-22 17:11:12. значение должно придти на момент открытия 2022-11-20 00:00:00)

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `31 день`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `5 бит`
- тип данных: `timedelta`

sync_time_days_ago

Количество целых дней с последней синхронизации.
Если стоят все 0 - то была синхронизация сегодня.Если стоят все 1 - то значение не валидно.

количество целых дней с последней синхронизации - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `7 дней`
- Лимитирование значения: `Значение принимает 0:00:00 если меньше и 7 days, 0:00:00 если больше`
- Переполнение значения: `Не допустимо`
- Размер: `3 бит`
- тип данных: `timedelta`

timestamp_s

Время в секундах от 2020-01-01 00:00:00.
Если стоят все 0 то время на приборе не ведется.

- Гранулярность данных: `1 секунда`
- Минимальное значение: `0 секунд`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 776 дней 17 часов 21 минута 3 секунды`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `26 бит`
- тип данных: `timedelta`

temperature

температура (с точностью до 1 градуса цельсия)

> Целое Беззнаковое

- Минимальное значение: `-35`
- Максимальное значение: `92`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `u7`

battery_volts

уровень заряда батареи (в вольтах)
замеряется в момент предыдущего сеанса отправки сообщения в эфир

уровень заряда батареи (в вольтах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `1`
- Минимальное значение: `0.0`
- Максимальное значение: `6.3`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `uf6p1`

event_reset

Была перезагрузка МК по какой то причине

- Размер: `1 бит`
- тип данных: `bool`

event_low_battery_level

Указание что батарейка разряжена

- Размер: `1 бит`
- тип данных: `bool`

event_temperature_limits

Низкая или высокая температура.

Выход за рамки эксплуатационных норм по температуре - Размер: `1 бит`

- тип данных: `bool`

direct_flow_volume

объем прямого потока (в м3, с 3мя знаками после запятой)
точное значение за весь период эксплуатации

объем прямого потока (в м3)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `3`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 4294967.295` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `32 бит`
- тип данных: `uf32p3`

direct_flow_volume_day_ago

разница объема прямого потока (в м3 с одним знаком после запятой) с окончанием предыдущего дня (относительно логики параметра days_ago)
значение дельты отправляемых в direct_flow_volume, полученное значение округляется в МЕНЬШУЮ сторону
00.0 - имеет значение ОТСУТСТВИЯ данных о предыдущем потреблении или дельта меньше чем 0.01 м3 (в расчет на верхнем уровне браться не должно).
12.7 - дельта слишком большая и не помещается, в отведенное в пакете, место (такой случай только для информации. В расчеты на верхнем уровне браться не должно)

разница объема прямого потока (в м3) 1 день назад

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `1`
- Минимальное значение: `0.0`

- Максимальное значение: `12.7`
- Лимитирование значения: `Значение принимает 0.0 если меньше и 12.7 если больше`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `uf7p1`

reverse_flow_volume

объем обратного потока (в м3)
точное значение за весь период эксплуатации

объем обратного потока (в м3)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 40.95`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `12 бит`
- тип данных: `uf12p2`

event_battery_warn

Первый порог низкого заряда батареи (синтетически задан этот уровень в приборе)

Предупреждение о низком заряде (1й порог) батареи - Размер: `1 бит`

- тип данных: `bool`

event_system_error

Указание что была внутренняя ошибка ПО. программа сама обработала сбой

- Размер: `1 бит`
- тип данных: `bool`

event_flow_reverse

Указание что обратный ход привысил допустимые лимиты за измеряемое время (для каждого прибора, специфичны размер и период измений)

Событие превышения лимита обратного хода - Размер: `1 бит`

- тип данных: `bool`

event_flow_speed_is_over_limit

Указание что было превышение лимитов скорости поотока

- Размер: `1 бит`
- тип данных: `bool`

event_sensor_error

Событие неисправности сенсора потока (например УЗ датчика)

- Размер: `1 бит`
- тип данных: `bool`

event_sensor_error_temperature

Указание что был сбой внешнего датчика температуры

- Размер: `1 бит`
- тип данных: `bool`

event_case_was_opened

Указание что было отрытие корпуса

- Размер: `1 бит`
- тип данных: `bool`

event_continuous_consumption

Событие непрерывного потребления (потенциальная утечка)

- Размер: `1 бит`
- тип данных: `bool`

event_no_resource

Событие нет ресурса (газа)

- Размер: `1 бит`
- тип данных: `bool`

event_magnet

Указание что магнит присутствует на момент отправки

- Размер: `1 бит`
- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDevWaterPackId.UL_DATA_16B__WATER_DAILY(515)
days_ago: 0.0s
sync_time_days_ago: 0.0s
timestamp_s: 33554431.0s
temperature: 23
battery_volts: 3.3
event_reset: false
event_low_battery_level: false
event_temperature_limits: true
direct_flow_volume: 112323.3
direct_flow_volume_day_ago: 3.5
reverse_flow_volume: 112323.3
event_battery_warn: true
event_system_error: false
event_flow_reverse: false
event_flow_speed_is_over_limit: false
event_sensor_error: false
event_sensor_error_temperature: false
event_case_was_opened: false
event_continuous_consumption: false
event_no_resource: true
event_magnet: false
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	3	0000011
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	0	00
packet_type_id.1.DFF	bool	1	1	1
packet_type_id.2.VAL	u2	2	1	01
packet_type_id.2.DFF	bool	1	0	0
days_ago	timedelta	5	0	00000
sync_time_days_ago	timedelta	3	0	000
timestamp_s	timedelta	26	33554431	01111111111111111111111111
temperature	u7	7	58	0111010
battery_volts	uf6p1	6	33	100001
event_reset	bool	1	0	0
event_low_battery_level	bool	1	0	0
event_temperature_limits	bool	1	1	1
direct_flow_volume	uf32p3	32	112323300	00000110101100011110101011100100
direct_flow_volume_day_ago	uf7p1	7	35	0100011
reverse_flow_volume	uf12p2	12	1098	010001001010
event_battery_warn	bool	1	1	1
event_system_error	bool	1	0	0
event_flow_reverse	bool	1	0	0
event_flow_speed_is_over_limit	bool	1	0	0
event_sensor_error	bool	1	0	0

name	type	size	value(int)	data(bits)
event_sensor_error_temperature	bool	1	0	0
event_case_was_opened	bool	1	0	0
event_continuous_consumption	bool	1	0	0
event_no_resource	bool	1	1	1
event_magnet	bool	1	0	0
RESERVED	u3	3	0	000

Result BIN (MSB - LSB):

```
00001000 00001010 00100101 00100011 00000110 10110001 11101010 11100100 10010000 10111010 01111111 11111111 11111111 11000000 00001100 10000011
```

Result HEX (little-endian):

```
83 0c c0 ff ff 7f ba 90 e4 ea b1 06 23 25 0a 08
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
import math
import typing


# PACKET (128 bits)   smpm_ul_device_water_meter_16b_daily
#
# RESULT int:       106865001590776975496165519715142237 47
# RESULT bin:  MSB  00001000 00001010 00100101 00100011 00000110 10110001 11101010 11100100 10010000 10111010 01111111 11111111 11111111 11000000 00001100 10000011  LSB
# RESULT hex:  LE   83 0c c0 ff ff 7f ba 90 e4 ea b1 06 23 25 0a 08
#
# name                          type      size  value(int)  data(bits)
# ----------------------------  --------  ----  ----------  ----------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL          u7           7          3  0000011
# packet_type_id.0.DFF          bool         1          1  1
# packet_type_id.1.VAL          u2           2          0  00
# packet_type_id.1.DFF          bool         1          1  1
# packet_type_id.2.VAL          u2           2          1  01
# packet_type_id.2.DFF          bool         1          0  0
# days_ago                      timedelta    5          0  00000
# sync_time_days_ago            timedelta    3          0  000
# timestamp_s                   timedelta   26   33554431  01111111111111111111111111
# temperature                   u7           7         58  0111010
# battery_volts                 uf6p1        6         33  100001
# event_reset                   bool         1          0  0
# event_low_battery_level       bool         1          0  0
# event_temperature_limits      bool         1          1  1
# direct_flow_volume            uf32p3      32  112323300  00000110101100011110101011100100
# direct_flow_volume_day_ago    uf7p1        7         35  0100011
# reverse_flow_volume           uf12p2      12       1098  010001001010
# event_battery_warn            bool         1          1  1
# event_system_error            bool         1          0  0
# event_flow_reverse            bool         1          0  0
# event_flow_speed_is_over_limit  bool       1          0  0
# event_sensor_error            bool         1          0  0
# event_sensor_error_temperature  bool       1          0  0
```

```
# event_case_was_opened          bool          1           0       0
# event_continuous_consumption    bool          1           0       0
# event_no_resource               bool          1           1     1
# event_magnet                    bool          1           0     0
# RESERVED                        u3            3           0  000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceWaterMeter16BDailyData(typing.NamedTuple):
    days_ago: timedelta
    sync_time_days_ago: timedelta
    timestamp_s: timedelta
    temperature: int
    battery_volts: float
    event_reset: bool
    event_low_battery_level: bool
    event_temperature_limits: bool
    direct_flow_volume: float
    direct_flow_volume_day_ago: float
    reverse_flow_volume: float
    event_battery_warn: bool
    event_system_error: bool
    event_flow_reverse: bool
    event_flow_speed_is_over_limit: bool
    event_sensor_error: bool
    event_sensor_error_temperature: bool
    event_case_was_opened: bool
    event_continuous_consumption: bool
    event_no_resource: bool
    event_magnet: bool

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((3) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((0) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
```

```
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 31
        result |= ((days_ago_tmp1) & (2 ** (5) - 1)) << size
        size += 5
        isinstance(data.sync_time_days_ago, (int, timedelta))
        result |= ((max(min(7, int(data.sync_time_days_ago.total_seconds() // 86400 if isinstance(data.sync_time_days_ago, timedelta) else data.sync_time_days_ago // 86400)), 0)) & (2 ** (3) - 1)) << size
        size += 3
        isinstance(data.timestamp_s, (int, timedelta))
        value_int_tmp2 = int(data.timestamp_s.total_seconds() // 1 if isinstance(data.timestamp_s, timedelta) else data.timestamp_s // 1) & 67108863
        result |= ((value_int_tmp2) & (2 ** (26) - 1)) << size
        size += 26
        assert isinstance(data.temperature, int)
        assert -35 <= data.temperature <= 92
        result |= (((data.temperature + 35)) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.battery_volts, (int, float))
        assert 0.0 <= data.battery_volts <= 6.3
        result |= ((int(round(float(data.battery_volts) * 10.0, 0))) & (2 ** (6) - 1)) << size
        size += 6
        assert isinstance(data.event_reset, bool)
        result |= ((int(data.event_reset)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_low_battery_level, bool)
        result |= ((int(data.event_low_battery_level)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_temperature_limits, bool)
        result |= ((int(data.event_temperature_limits)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.direct_flow_volume, (int, float))
        result |= ((int(round(float(data.direct_flow_volume) * 1000.0, 0)) & 4294967295) & (2 ** (32) - 1)) << size
        size += 32
        assert isinstance(data.direct_flow_volume_day_ago, (int, float))
        result |= ((int(round(max(min(12.7, float(data.direct_flow_volume_day_ago)), 0.0) * 10.0, 0))) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.reverse_flow_volume, (int, float))
        result |= ((int(round(float(data.reverse_flow_volume) * 100.0, 0)) & 4095) & (2 ** (12) - 1)) << size
        size += 12
        assert isinstance(data.event_battery_warn, bool)
        result |= ((int(data.event_battery_warn)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_system_error, bool)
        result |= ((int(data.event_system_error)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_flow_reverse, bool)
        result |= ((int(data.event_flow_reverse)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_flow_speed_is_over_limit, bool)
        result |= ((int(data.event_flow_speed_is_over_limit)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_sensor_error, bool)
        result |= ((int(data.event_sensor_error)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_sensor_error_temperature, bool)
        result |= ((int(data.event_sensor_error_temperature)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_case_was_opened, bool)
        result |= ((int(data.event_case_was_opened)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_continuous_consumption, bool)
        result |= ((int(data.event_continuous_consumption)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_no_resource, bool)
```

```
        result |= ((int(data.event_no_resource)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_magnet, bool)
        result |= ((int(data.event_magnet)) & (2 ** (1) - 1)) << size
        size += 1
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceWaterMeter16BDailyData':
        result__el_tmp3: typing.Dict[str, typing.Any] = dict()
        if 3 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp3["days_ago"] = timedelta(seconds=buf.shift(5) * 86400)
        result__el_tmp3["sync_time_days_ago"] = timedelta(seconds=buf.shift(3) * 86400)
        result__el_tmp3["timestamp_s"] = timedelta(seconds=buf.shift(26) * 1)
        result__el_tmp3["temperature"] = buf.shift(7) + -35
        result__el_tmp3["battery_volts"] = round(buf.shift(6) / 10.0, 1)
        result__el_tmp3["event_reset"] = bool(buf.shift(1))
        result__el_tmp3["event_low_battery_level"] = bool(buf.shift(1))
        result__el_tmp3["event_temperature_limits"] = bool(buf.shift(1))
        result__el_tmp3["direct_flow_volume"] = round(buf.shift(32) / 1000.0, 3)
        result__el_tmp3["direct_flow_volume_day_ago"] = round(buf.shift(7) / 10.0, 1)
        result__el_tmp3["reverse_flow_volume"] = round(buf.shift(12) / 100.0, 2)
        result__el_tmp3["event_battery_warn"] = bool(buf.shift(1))
        result__el_tmp3["event_system_error"] = bool(buf.shift(1))
        result__el_tmp3["event_flow_reverse"] = bool(buf.shift(1))
        result__el_tmp3["event_flow_speed_is_over_limit"] = bool(buf.shift(1))
        result__el_tmp3["event_sensor_error"] = bool(buf.shift(1))
        result__el_tmp3["event_sensor_error_temperature"] = bool(buf.shift(1))
        result__el_tmp3["event_case_was_opened"] = bool(buf.shift(1))
        result__el_tmp3["event_continuous_consumption"] = bool(buf.shift(1))
        result__el_tmp3["event_no_resource"] = bool(buf.shift(1))
        result__el_tmp3["event_magnet"] = bool(buf.shift(1))
        result = SmpmUlDeviceWaterMeter16BDailyData(**result__el_tmp3)
        buf.shift(3)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Воды] Пакет дневных показаний для счетчиков воды с запорным клапаном и маской событий (8 байт)

Тип пакета: `Uplink`

Размер пакета: `64 бит (8байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_water_meter_08b_valve_daily`

Описание

Пакет отправляется каждый день дважды (на разных каналах). Первый раз - в промежутке с 00:00-12:00. Второй с 12:00-00:00

Наличие конкретного события (за период времени от последнего дневого пакета) указывается одним единичным битом

Пакет с экстренным событием (неисправность УЗ датчиков, низкий заряд (оба порога), Температура, выходящая за пределы) должно отсылаться сразу как это произошло.

Событие ресета отправляется сразу (спустя установленную задержку) после того как перезагрузился. Отправляется оно каждый раз, как перезагружается. Не чаще 3х раз за день

Флаг будет отправляться активным с каждым пакетом до тех пор, пока воздействие будет активно

Расчет значений за предыдущие дни (например сейчас 2022-12-16 20:13:03)

Пакет может быть отправлен только в рамках температурного режима от +5 градусов Цельсия и выше

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	478 (*bin* **001 11011110**)
direct_flow_volume	объем прямого потока (в м3)	32 бит	uf32p3	0.0	-	4294967.295	-
battery_voltage	минимальное напряжение батареи за сутки (в вольтах)	8 бит	uf8p2	0.0	2.55	-	-
event_temperature_is_over_limit	Выход за рамки эксплуатационных норм по температуре	1 бит	bool	0	1	-	-
event_low_battery	Низкое напряжение батареи	1 бит	bool	0	1	-	-
event_no_resource	Отсутствие воды	1 бит	bool	0	1	-	-
event_ultrasonic_error	Ошибка УЗ	1 бит	bool	0	1	-	-
event_leakage	Утечка	1 бит	bool	0	1	-	-
event_breach	Прорыв	1 бит	bool	0	1	-	-
event_tampering	Вскрытие устройства	1 бит	bool	0	1	-	-
event_reset	Системные ошибки Ресет	1 бит	bool	0	1	-	-
event_shutoff_valve_switch	Запорный клапан закрыт (активен)	1 бит	bool	0	1	-	-
event_shutoff_valve_switch_error	Ошибка активации (закрытия) запорного клапана	1 бит	bool	0	1	-	-
RESERVED	-	3 бит	-	-	-	-	

Референсные значения типов данных

direct_flow_volume

объем прямого потока (в м3, с 3-мя знаками после запятой)
точное значение за весь период эксплуатации

объем прямого потока (в м3)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `3`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 4294967.295` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `32 бит`
- тип данных: `uf32p3`

battery_voltage

уровень заряда батареи (в вольтах) - 1.8 вольт
минимальное напряжение батареи за сутки

минимальное напряжение батареи за сутки (в вольтах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `2.55`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `8 бит`
- тип данных: `uf8p2`

event_temperature_is_over_limit

Низкая или высокая температура.

Выход за рамки эксплуатационных норм по температуре - Размер: `1 бит`

- тип данных: `bool`

event_low_battery

Низкое напряжение батареи

- Размер: `1 бит`
- тип данных: `bool`

event_no_resource

Отсутствие воды

- Размер: `1 бит`
- тип данных: `bool`

event_ultrasonic_error

Ошибка УЗ

- Размер: `1 бит`
- тип данных: `bool`

event_leakage

Утечка

- Размер: `1 бит`
- тип данных: `bool`

event_breach

Прорыв

- Размер: `1 бит`
- тип данных: `bool`

event_tampering

Вскрытие устройства

- Размер: `1 бит`
- тип данных: `bool`

event_reset

Системные ошибки Ресет

- Размер: `1 бит`
- тип данных: `bool`

event_shutoff_valve_switch

Запорный клапан закрыт (активен)

- Размер: `1 бит`

- тип данных: `bool`

event_shutoff_valve_switch_error

Ошибка активации (закрытия) запорного клапана

- Размер: `1 бит`
- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDevWaterPackId.UL_DATA_08B__WATER_VALVE_DAILY(222)
direct_flow_volume: 112323.3
battery_voltage: 2.0
event_temperature_is_over_limit: true
event_low_battery: false
event_no_resource: false
event_ultrasonic_error: false
event_leakage: true
event_breach: true
event_tampering: true
event_reset: true
event_shutoff_valve_switch: false
event_shutoff_valve_switch_error: false
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	94	1011110
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	1	01
packet_type_id.1.DFF	bool	1	0	0
direct_flow_volume	uf32p3	32	112323300	00000110101100011110101011100100
battery_voltage	uf8p2	8	200	11001000
event_temperature_is_over_limit	bool	1	1	1
event_low_battery	bool	1	0	0
event_no_resource	bool	1	0	0
event_ultrasonic_error	bool	1	0	0
event_leakage	bool	1	1	1
event_breach	bool	1	1	1
event_tampering	bool	1	1	1
event_reset	bool	1	1	1
event_shutoff_valve_switch	bool	1	0	0
event_shutoff_valve_switch_error	bool	1	0	0
RESERVED	u3	3	0	000

Result BIN (MSB - LSB):

```
00000111 10001110 01000000 00110101 10001111 01010111 00100001 11011110
```

Result HEX (little-endian):

```
de 21 57 8f 35 40 8e 07
```

Парсинг/Сериализация значений

Python C

```python
import math
import typing


# PACKET (64 bits)   smpn_ul_device_water_meter_08b_valve_daily
#
# RESULT int:        544443203740705246
# RESULT bin:  MSB   00000111 10001110 01000000 00110101 10001111 01010111 00100001 11011110   LSB
# RESULT hex:  LE    de 21 57 8f 35 40 8e 07
#
# name                              type    size  value(int)                                                  data(bits)
# --------------------------------  ------  ----  ----------  ----------------------------------------------------------------
# packet_type_id.0.VAL              u7         7          94                                                     1011110
# packet_type_id.0.DFF              bool       1           1                                                    1
# packet_type_id.1.VAL              u2         2           1                                                   01
# packet_type_id.1.DFF              bool       1           0                                                  0
# direct_flow_volume                uf32p3    32   112323300                    00000110101100011110101011100100
# battery_voltage                   uf8p2      8         200              11001000
# event_temperature_is_over_limit   bool       1           1             1
# event_low_battery                 bool       1           0            0
# event_no_resource                 bool       1           0           0
# event_ultrasonic_error            bool       1           0          0
# event_leakage                     bool       1           1         1
# event_breach                      bool       1           1        1
# event_tampering                   bool       1           1       1
# event_reset                       bool       1           1      1
# event_shutoff_valve_switch        bool       1           0      0
# event_shutoff_valve_switch_error  bool       1           0     0
# RESERVED                          u3         3           0  000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res
```

```
    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceWaterMeter08BValveDailyData(typing.NamedTuple):
    direct_flow_volume: float
    battery_voltage: float
    event_temperature_is_over_limit: bool
    event_low_battery: bool
    event_no_resource: bool
    event_ultrasonic_error: bool
    event_leakage: bool
    event_breach: bool
    event_tampering: bool
    event_reset: bool
    event_shutoff_valve_switch: bool
    event_shutoff_valve_switch_error: bool

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((94) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.direct_flow_volume, (int, float))
        result |= ((int(round(float(data.direct_flow_volume) * 1000.0, 0)) & 4294967295) & (2 ** (32) - 1)) << size
        size += 32
        assert isinstance(data.battery_voltage, (int, float))
        assert 0.0 <= data.battery_voltage <= 2.55
        result |= ((int(round(float(data.battery_voltage) * 100.0, 0))) & (2 ** (8) - 1)) << size
        size += 8
        assert isinstance(data.event_temperature_is_over_limit, bool)
        result |= ((int(data.event_temperature_is_over_limit)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_low_battery, bool)
        result |= ((int(data.event_low_battery)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_no_resource, bool)
        result |= ((int(data.event_no_resource)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_ultrasonic_error, bool)
        result |= ((int(data.event_ultrasonic_error)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_leakage, bool)
        result |= ((int(data.event_leakage)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_breach, bool)
        result |= ((int(data.event_breach)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_tampering, bool)
        result |= ((int(data.event_tampering)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_reset, bool)
        result |= ((int(data.event_reset)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_shutoff_valve_switch, bool)
        result |= ((int(data.event_shutoff_valve_switch)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_shutoff_valve_switch_error, bool)
        result |= ((int(data.event_shutoff_valve_switch_error)) & (2 ** (1) - 1)) << size
```

```
        size += 1
        return result.to_bytes(8, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceWaterMeter08BValveDailyData':
        result__el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 94 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp1["direct_flow_volume"] = round(buf.shift(32) / 1000.0, 3)
        result__el_tmp1["battery_voltage"] = round(buf.shift(8) / 100.0, 2)
        result__el_tmp1["event_temperature_is_over_limit"] = bool(buf.shift(1))
        result__el_tmp1["event_low_battery"] = bool(buf.shift(1))
        result__el_tmp1["event_no_resource"] = bool(buf.shift(1))
        result__el_tmp1["event_ultrasonic_error"] = bool(buf.shift(1))
        result__el_tmp1["event_leakage"] = bool(buf.shift(1))
        result__el_tmp1["event_breach"] = bool(buf.shift(1))
        result__el_tmp1["event_tampering"] = bool(buf.shift(1))
        result__el_tmp1["event_reset"] = bool(buf.shift(1))
        result__el_tmp1["event_shutoff_valve_switch"] = bool(buf.shift(1))
        result__el_tmp1["event_shutoff_valve_switch_error"] = bool(buf.shift(1))
        result = SmpmUlDeviceWaterMeter08BValveDailyData(**result__el_tmp1)
        buf.shift(3)
        return result
```

Последнее обновление: 2024-09-16

[Счетчики Импульсов] Пакет текущих показаний объёма для счетчиков импульсов (12байт)

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_jupiter_12b_counter_volume`

Описание

Отправляется каждый день. Дополнительно дублируется пакет на другом канале для повышения уровня собираемости. Последние 32 байта использовать нельзя, они используются из-за ограничений алгоритма шифрования XTEA.

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	469 (*bin* **001 11010101**)
volume_channel_1	Объём по каналу 1 (с учетом множителя и коэфицента пересчёта)	32 бит	uf32p3	0.0	-	4294967.295	-
volume_channel_2	Объём по каналу 2 (с учетом множителя и коэфицента пересчёта)	32 бит	uf32p3	0.0	-	4294967.295	-
battery_volts	минимальное напряжение батареи за сутки (в вольтах)	8 бит	uf8p2	0.0	2.55	-	-
temperature	температура (с точностью до 1 градуса цельсия)	7 бит	u7	-35	92	-	-
event_reset	Событие сброса МК по любой причине	1 бит	bool	0	1	-	-
event_low_battery_level	Brown-out reset (BOR) flag	1 бит	bool	0	1	-	-
event_low_ambient_temperature	Событие низкой температуры окружающей среды	1 бит	bool	0	1	-	-
RESERVED	-	35 бит	-	-	-	-	

Референсные значения типов данных

volume_channel_1

Объём по каналу 1 (с учетом множителя и коэфицента пересчёта)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `3`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 4294967.295` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `32 бит`
- тип данных: `uf32p3`

volume_channel_2

Объём по каналу 2 (с учетом множителя и коэфицента пересчёта)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `3`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 4294967.295` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `32 бит`
- тип данных: `uf32p3`

battery_volts

уровень заряда батареи (в вольтах) - 1.8 вольт
минимальное напряжение батареи за сутки

минимальное напряжение батареи за сутки (в вольтах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `2.55`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `8 бит`
- тип данных: `uf8p2`

temperature

температура (с точностью до 1 градуса цельсия)

> Целое Беззнаковое

- Минимальное значение: `-35`
- Максимальное значение: `92`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `u7`

event_reset

Событие сброса МК по любой причине

- Размер: `1 бит`
- тип данных: `bool`

event_low_battery_level

Brown-out reset (BOR) flag
Перезагрузка по питанию (внештатная, просадка по напряжению или кто-то вскрыл прибор и отключил батарейку. передача пакета на БС при разряженной батарейке или очень низкой температуре.)

Brown-out reset (BOR) flag - Размер: `1 бит`

- тип данных: `bool`

event_low_ambient_temperature

Событие низкой температуры окружающей среды

- Размер: `1 бит`
- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDevJupiterPackId.UL_DATA_12B__JUPITER_COUNTER_VOLUME(213)
volume_channel_1: 2147483.647
volume_channel_2: 2147483.647
battery_volts: 2.0
temperature: 23
event_reset: true
event_low_battery_level: true
event_low_ambient_temperature: true
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	85	1010101
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	1	01
packet_type_id.1.DFF	bool	1	0	0
volume_channel_1	uf32p3	32	2147483647	01111111111111111111111111111111
volume_channel_2	uf32p3	32	2147483647	01111111111111111111111111111111
battery_volts	uf8p2	8	200	11001000
temperature	u7	7	58	0111010
event_reset	bool	1	1	1
event_low_battery_level	bool	1	1	1
event_low_ambient_temperature	bool	1	1	1
RESERVED	u35	35	0	00000000000000000000000000000000000

Result BIN (MSB - LSB):

```
00000000 00000000 00000000 00000000 00011101 11010110 01000011 11111111 11111111 11111111 11111011 11111111 11111111 11111111 11111001 11010101
```

Result HEX (little-endian):

```
d5 f9 ff ff ff fb ff ff ff 43 d6 1d 00 00 00 00
```

Парсинг/Сериализация значений

Python   C

```python
import math
import typing


# PACKET (128 bits)   smpm_ul_device_jupiter_12b_counter_volume
#
# RESULT int:        9234096531137368374233790933
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00011101 11010110 01000011 11111111 11111111 11111111 11111011 11111111 11111111 11111111 11111001 11010101   LSB
# RESULT hex:  LE    d5 f9 ff ff ff fb ff ff ff 43 d6 1d 00 00 00 00
#
# name                           type    size  value(int)                                                                                                                        data(bits)
# -----------------------------  ------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL           u7         7          85                                                                                                                           1010101
# packet_type_id.0.DFF           bool       1           1                                                                                                                          1
# packet_type_id.1.VAL           u2         2           1                                                                                                                        01
# packet_type_id.1.DFF           bool       1           0                                                                                                                       0
# volume_channel_1               uf32p3    32  2147483647                                                                                     01111111111111111111111111111111
# volume_channel_2               uf32p3    32  2147483647                                                     01111111111111111111111111111111
# battery_volts                  uf8p2      8         200                                             11001000
# temperature                    u7         7          58                                      0111010
# event_reset                    bool       1           1                                     1
# event_low_battery_level        bool       1           1                                    1
# event_low_ambient_temperature  bool       1           1                                   1
# RESERVED                       u35       35           0  00000000000000000000000000000000000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size
```

```
class SmpmUlDeviceJupiter12BCounterVolumeData(typing.NamedTuple):
    volume_channel_1: float
    volume_channel_2: float
    battery_volts: float
    temperature: int
    event_reset: bool
    event_low_battery_level: bool
    event_low_ambient_temperature: bool

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((85) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.volume_channel_1, (int, float))
        result |= ((int(round(float(data.volume_channel_1) * 1000.0, 0)) & 4294967295) & (2 ** (32) - 1)) << size
        size += 32
        assert isinstance(data.volume_channel_2, (int, float))
        result |= ((int(round(float(data.volume_channel_2) * 1000.0, 0)) & 4294967295) & (2 ** (32) - 1)) << size
        size += 32
        assert isinstance(data.battery_volts, (int, float))
        assert 0.0 <= data.battery_volts <= 2.55
        result |= ((int(round(float(data.battery_volts) * 100.0, 0))) & (2 ** (8) - 1)) << size
        size += 8
        assert isinstance(data.temperature, int)
        assert -35 <= data.temperature <= 92
        result |= (((data.temperature + 35)) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.event_reset, bool)
        result |= ((int(data.event_reset)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_low_battery_level, bool)
        result |= ((int(data.event_low_battery_level)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.event_low_ambient_temperature, bool)
        result |= ((int(data.event_low_ambient_temperature)) & (2 ** (1) - 1)) << size
        size += 1
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceJupiter12BCounterVolumeData':
        result_el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 85 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result_el_tmp1["volume_channel_1"] = round(buf.shift(32) / 1000.0, 3)
        result_el_tmp1["volume_channel_2"] = round(buf.shift(32) / 1000.0, 3)
        result_el_tmp1["battery_volts"] = round(buf.shift(8) / 100.0, 2)
        result_el_tmp1["temperature"] = buf.shift(7) + -35
        result_el_tmp1["event_reset"] = bool(buf.shift(1))
        result_el_tmp1["event_low_battery_level"] = bool(buf.shift(1))
        result_el_tmp1["event_low_ambient_temperature"] = bool(buf.shift(1))
        result = SmpmUlDeviceJupiter12BCounterVolumeData(**result_el_tmp1)
```

```
        buf.shift(35)
        return result
```

Последнее обновление: 2024-06-12

[Общий] Пакет ответа на даунлинк запрос

Тип пакета: `Uplink`

Размер пакета: `64 бит (8байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_dl_answer`

Описание

Пакет отправляется в ответ на даунлинк пакет для указания что следущие N пакетов будут ответом на запрос.

Необходимость ответа этим пакетом определяется договоренностью между прибором и верхним уровнем.

Пакеты ответа обязаны быть в одном пэйлоаде uplink

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	8 бит	const(dff(8,3,3,3...))	-	-	-	3 (*hex* **03**)
downlink_packet_id	ID пакета запроса на который отвечает прибор	16 бит	SmpmUlDeviceDlAnswerDataDownlinkPacketId	SmpmUlDeviceDlAnswerDataDownlinkPacketId.GET_ECHO	SmpmUlDeviceDlAnswerDataDownlinkPacketId.SET_RELAY	-	-
downlink_packet_crc	CRC32 downlink SMP-M пакета на который отвечает прибор	32 бит	u32	0	4294967295	-	-
answer_packets_count	Количество пакетов, следуемых за этим пакетов, которые является ответом на запрос по даунлинк	4 бит	u4	0	15	-	-
RESERVED	-	4 бит	-	-	-	-	

Референсные значения типов данных

downlink_packet_id

ID пакета запроса на который отвечает прибор

- Минимальное значение: `SmpmUlDeviceDlAnswerDataDownlinkPacketId.GET_ECHO`
- Максимальное значение: `SmpmUlDeviceDlAnswerDataDownlinkPacketId.SET_RELAY`
- Размер: `16 бит`
- тип данных: `SmpmUlDeviceDlAnswerDataDownlinkPacketId`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
1	GET_ECHO

Значение	Имя
2	SET_CLOCK
128	GET_DATA_SHORT
129	GET_DATA_LONG
150	SET_REGULAR_DATA_SENDING
170	SET_RELAY

downlink_packet_crc

CRC32 downlink SMP-M пакета на который отвечает прибор

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `4294967295`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `32 бит`
- тип данных: `u32`

answer_packets_count

Количество пакетов, следуемых за этим пакетов, которые является ответом на запрос по даунлинк

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `15`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `4 бит`
- тип данных: `u4`

Пример пакета

```
packet_type_id: UpLinkDevId.DOWNLINK_ANSWER(3)
downlink_packet_id: SmpmUlDeviceDlAnswerDataDownlinkPacketId.GET_ECHO(1)
downlink_packet_crc: 2147483647
answer_packets_count: 7
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	3	`0000011`
packet_type_id.0.DFF	bool	1	0	`0`

name	type	size	value(int)	data(bits)
downlink_packet_id	SmpmUlDeviceDlAnswerDataDownlinkPacketId	16	1	0000000000000001
downlink_packet_crc	u32	32	2147483647	01111111111111111111111111111111
answer_packets_count	u4	4	7	0111
RESERVED	u4	4	0	0000

Result BIN (MSB - LSB):

```
00000111 01111111 11111111 11111111 11111111 00000000 00000001 00000011
```

Result HEX (little-endian):

```
03 01 00 ff ff ff 7f 07
```

Парсинг/Сериализация значений

Python C

```python
from enum import IntEnum, unique
import math
import typing


# PACKET (64 bits)   smpm_ul_device_dl_answer
#
# RESULT int:        540431955267682563
# RESULT bin:  MSB   00000111 01111111 11111111 11111111 11111111 00000000 00000001 00000011   LSB
# RESULT hex:  LE    03 01 00 ff ff ff 7f 07
#
# name                 type                                  size  value(int)                                          data(bits)
# -------------------  ------------------------------------  ----  ----------  ------------------------------------------------------------
# packet_type_id.0.VAL  u7                                     7          3                                                    0000011
# packet_type_id.0.DFF  bool                                   1          0                                                          0
# downlink_packet_id    SmpmUlDeviceDlAnswerDataDownlinkPacketId    16          1                                  0000000000000001
# downlink_packet_crc   u32                                   32  2147483647         01111111111111111111111111111111
# answer_packets_count  u4                                     4          7     0111
# RESERVED              u4                                     4          0  0000
@unique
class SmpmUlDeviceDlAnswerDataDownlinkPacketId(IntEnum):
    GET_ECHO = 1
    SET_CLOCK = 2
    GET_DATA_SHORT = 128
    GET_DATA_LONG = 129
    SET_REGULAR_DATA_SENDING = 150
    SET_RELAY = 170


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
```

```
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceDlAnswerData(typing.NamedTuple):
    downlink_packet_id: SmpmUlDeviceDlAnswerDataDownlinkPacketId
    downlink_packet_crc: int
    answer_packets_count: int

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((3) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.downlink_packet_id, SmpmUlDeviceDlAnswerDataDownlinkPacketId)
        result |= ((data.downlink_packet_id.value) & (2 ** (16) - 1)) << size
        size += 16
        assert isinstance(data.downlink_packet_crc, int)
        assert 0 <= data.downlink_packet_crc <= 4294967295
        result |= ((data.downlink_packet_crc) & (2 ** (32) - 1)) << size
        size += 32
        assert isinstance(data.answer_packets_count, int)
        assert 0 <= data.answer_packets_count <= 15
        result |= ((data.answer_packets_count) & (2 ** (4) - 1)) << size
        size += 4
        return result.to_bytes(8, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceDlAnswerData':
        result__el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 3 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp1["downlink_packet_id"] = SmpmUlDeviceDlAnswerDataDownlinkPacketId(buf.shift(16))
        result__el_tmp1["downlink_packet_crc"] = buf.shift(32) + 0
        result__el_tmp1["answer_packets_count"] = buf.shift(4) + 0
        result = SmpmUlDeviceDlAnswerData(**result__el_tmp1)
        buf.shift(4)
        return result
```

Последнее обновление: 2023-11-30

[Счетчики Электроэнергии] Пакет показаний энергии с ретроспективой показаний и за последние 3 дня/месяца, общее или по тарифу

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_retrospective_energy`

Описание

Пакет самых частых тарифов с ретроспективой для энергии за текущее показание и за последние 3 дня

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id_enum	идентификатор пакета	11 бит	dff(8,3,3,3...)	-	-	-	-
↳ VAL	segment value	[7, 2] бит	-	0	127	-	-
↳ DFF	has next segment flag	1 бит	bool	0	1	-	-
is_valid	Статус текущих данных: 1 – хорошо 0 – проблема (ошибка самотестирования, квитируемые со-бытия, реле отключено).	1 бит	bool	0	1	-	-
period_ago	Количество дней/месяцев от текущего дня/месяца	5 бит	u5	0	31	-	-
value_current	текущее значение тарифа (в киловаттах).	27 бит	uf27p2	0.0	-	1342177.27	-
value_previous_1_delta	значение тарифа день/месяц назад (в киловаттах).	24 бит	uf24p2	0.0	-	167772.15	-
value_previous_2_delta	значение тарифа два дня/месяца назад (в киловаттах).	24 бит	uf24p2	0.0	-	167772.15	-
value_previous_3_delta	значение тарифа три дня/месяца назад (в киловаттах).	24 бит	uf24p2	0.0	-	167772.15	-
RESERVED	-	12 бит	-	-	-	-	

Референсные значения типов данных

packet_type_id_enum

идентификатор пакета

- Размер: `11 бит`
- тип данных: `dff(8,3,3,3...)`

Алгоритм запаковки:

- переданное значение разбивается битами на группы `VAL` (начиная с минимального).
- каждая группа имеет флаг `DFF` **(Dynamic Field Flag)**, он является последним по порядку в VAL-группе
- Флаг `DFF` определяет наличие следующей группы (если 1 то следующая группа существует, если 0 то это последняя группа)

Размеры группы VAL:

№ группы	размер группы (бит)
1	8
2	3
3	3
4	3
...далее	3

Перечисление типов пакетов профилей по типу энергии и тарифу

Может принимать одно значение из приведенного ниже списка:

Значение	Имя	преобразованные биты(LE)
400	DAILY_ENERGY_ACTIVE_CONSUMED	`0 11 1 0010000`
401	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1	`0 11 1 0010001`
402	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2	`0 11 1 0010010`
403	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3	`0 11 1 0010011`
404	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4	`0 11 1 0010100`
405	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM	`0 11 1 0010101`
406	DAILY_ENERGY_REACTIVE_CONSUMED	`0 11 1 0010110`
407	DAILY_ENERGY_ACTIVE_GENERATED	`0 11 1 0010111`
408	DAILY_ENERGY_REACTIVE_GENERATED	`0 11 1 0011000`
409	MONTHLY_ENERGY_ACTIVE_CONSUMED	`0 11 1 0011001`
410	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1	`0 11 1 0011010`
411	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2	`0 11 1 0011011`
412	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3	`0 11 1 0011100`
413	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4	`0 11 1 0011101`
414	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM	`0 11 1 0011110`
415	MONTHLY_ENERGY_REACTIVE_CONSUMED	`0 11 1 0011111`
416	MONTHLY_ENERGY_ACTIVE_GENERATED	`0 11 1 0100000`
417	MONTHLY_ENERGY_REACTIVE_GENERATED	`0 11 1 0100001`

is_valid

Статус текущих данных: 1 – хорошо 0 – проблема (ошибка самотестирования, квитируемые со-бытия, реле отключено).
В счетчике статус индивидуален для каждого часа/су-ток/месяца,
то в данной команде передается статус за теку-щие сутки (если хотя бы один час плохой, то статус плохой)

- Размер: `1 бит`
- тип данных: `bool`

period_ago

Количество дней/месяцев от текущего дня/месяца
Пример: пакет за 3 последних дня, сегодня 2022-12-31

period_ago=0
value_current - показания за конец дня 2022-12-31
value_previous_1_delta - дельта за конец дня 2022-12-30
value_previous_2_delta - дельта за конец дня 2022-12-29
value_previous_3_delta - дельта за конец дня 2022-12-28
period_ago=1
value_current - показания за конец дня 2022-12-30
value_previous_1_delta - дельта за конец дня 2022-12-29
value_previous_2_delta - дельта за конец дня 2022-12-28
value_previous_3_delta - дельта за конец дня 2022-12-27
пакет за 3 последних месяца
period_ago=0
value_current - показания за конец дня 2022-12-31
value_previous_1_delta - дельта за конец дня 2022-11-30
value_previous_2_delta - дельта за конец дня 2022-10-31
value_previous_3_delta - дельта за конец дня 2022-09-30
сегодня 2022-12-31
period_ago=1
value_current - показания за конец дня 2022-11-30
value_previous_1_delta - дельта за конец дня 2022-10-31
value_previous_2_delta - дельта за конец дня 2022-09-30
value_previous_3_delta - дельта за конец дня 2022-08-31

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `31`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `5 бит`
- тип данных: `u5`

value_current

текущее значение тарифа (в киловаттах).
всегда округляются в меньшую сторону

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 1342177.27`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `27 бит`
- тип данных: `uf27p2`

value_previous_1_delta

значение тарифа день/месяц назад (в киловаттах).
всегда округляются в меньшую сторону

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 167772.15`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `24 бит`
- тип данных: `uf24p2`

value_previous_2_delta

значение тарифа два дня/месяца назад (в киловаттах).
всегда округляются в меньшую сторону

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 167772.15`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `24 бит`
- тип данных: `uf24p2`

value_previous_3_delta

значение тарифа три дня/месяца назад (в киловаттах).
всегда округляются в меньшую сторону

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 167772.15`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `24 бит`
- тип данных: `uf24p2`

Пример пакета

```
packet_type_id_enum: SmpmUlDeviceEnergy16bRetrospectiveEnergyIds.DAILY_ENERGY_ACTIVE_CONSUMED(400)
is_valid: false
period_ago: 15
value_current: 4312123.0
value_previous_1_delta: 5413.0
value_previous_2_delta: 0.0
value_previous_3_delta: 123123.0
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id_enum.0.VAL	u7	7	16	0010000
packet_type_id_enum.0.DFF	bool	1	1	1
packet_type_id_enum.1.VAL	u2	2	3	11
packet_type_id_enum.1.DFF	bool	1	0	0
is_valid	bool	1	0	0
period_ago	u5	5	15	01111
value_current	uf27p2	27	28559116	001101100111100011100001100
value_previous_1_delta	uf24p2	24	541300	000010000100001001110100
value_previous_2_delta	uf24p2	24	0	000000000000000000000000
value_previous_3_delta	uf24p2	24	12312300	101110111101111011101100
RESERVED	u12	12	0	000000000000

Result BIN (MSB - LSB):

```
00000000 00001011 10111101 11101110 11000000 00000000 00000000 00000000 10000100 00100111 01000011 01100111 10001110 00011000 11110011 10010000
```

Result HEX (little-endian):

```
90 f3 18 8e 67 43 27 84 00 00 00 c0 ee bd 0b 00
```

Парсинг/Сериализация значений

Python C

```python
from enum import IntEnum, unique
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_retrospective_energy
#
# RESULT int:        60967556582773559757215997875516304
# RESULT bin:  MSB   00000000 00001011 10111101 11101110 11000000 00000000 00000000 00000000 10000100 00100111 01000011 01100111 10001110 00011000 11110011 10010000   LSB
# RESULT hex:  LE    90 f3 18 8e 67 43 27 84 00 00 00 c0 ee bd 0b 00
#
# name                        type    size  value(int)                                                                                              data(bits)
# --------------------------  ------  ----  ----------  ----------------------------------------------------------------------------------------------------------
```

```
# packet_type_id_enum.0.VAL  u7        7         16                                                                                                  0010000
# packet_type_id_enum.0.DFF  bool      1          1                                                                                                 1
# packet_type_id_enum.1.VAL  u2        2          3                                                                                               11
# packet_type_id_enum.1.DFF  bool      1          0                                                                                              0
# is_valid                   bool      1          0                                                                                             0
# period_ago                 u5        5         15                                                                                    01111
# value_current              uf27p2   27   28559116                                                                  00110110011110001110000l100
# value_previous_1_delta     uf24p2   24     541300                                                      000010000100001001110100
# value_previous_2_delta     uf24p2   24          0                                          000000000000000000000000
# value_previous_3_delta     uf24p2   24   12312300                  101110111101111011101100
# RESERVED                   u12      12          0  000000000000
@unique
class SmpmUlDeviceEnergy16bRetrospectiveEnergyIds(IntEnum):
    DAILY_ENERGY_ACTIVE_CONSUMED = 400
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 401
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 402
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 403
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 404
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 405
    DAILY_ENERGY_REACTIVE_CONSUMED = 406
    DAILY_ENERGY_ACTIVE_GENERATED = 407
    DAILY_ENERGY_REACTIVE_GENERATED = 408
    MONTHLY_ENERGY_ACTIVE_CONSUMED = 409
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 410
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 411
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 412
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 413
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 414
    MONTHLY_ENERGY_REACTIVE_CONSUMED = 415
    MONTHLY_ENERGY_ACTIVE_GENERATED = 416
    MONTHLY_ENERGY_REACTIVE_GENERATED = 417


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16BRetrospectiveEnergyData(typing.NamedTuple):
    packet_type_id_enum: SmpmUlDeviceEnergy16bRetrospectiveEnergyIds
    is_valid: bool
    period_ago: int
    value_current: float
    value_previous_1_delta: float
    value_previous_2_delta: float
    value_previous_3_delta: float
```

```python
    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        packet_type_id_enum__value_int_tmp1 = 0
        packet_type_id_enum__value_size_tmp2 = 0
        assert isinstance(data.packet_type_id_enum, SmpmUlDeviceEnergy16bRetrospectiveEnergyIds)
        packet_type_id_enum__value_int_tmp1 |= ((data.packet_type_id_enum.value) & (2 ** (11) - 1)) << packet_type_id_enum__value_size_tmp2
        packet_type_id_enum__value_size_tmp2 += 11
        packet_type_id_enum__max_size_tmp3 = 0
        packet_type_id_enum__steps_tmp4 = [8,3,3,3]
        for packet_type_id_enum__j_tmp5 in range(32):
            packet_type_id_enum__step_tmp6 = packet_type_id_enum__steps_tmp4[packet_type_id_enum__j_tmp5] if packet_type_id_enum__j_tmp5 < len(packet_type_id_enum__steps_tmp4) else packet_type_id_enum__steps_tmp4[-1]
            packet_type_id_enum__max_size_tmp3 += packet_type_id_enum__step_tmp6
            packet_type_id_enum__current_part_value_tmp7 = packet_type_id_enum__value_int_tmp1 & (2 ** packet_type_id_enum__step_tmp6 - 1)
            packet_type_id_enum__value_int_tmp1 = packet_type_id_enum__value_int_tmp1 >> (packet_type_id_enum__step_tmp6 - 1)
            result |= ((packet_type_id_enum__current_part_value_tmp7) & (2 ** ((packet_type_id_enum__step_tmp6 - 1)) - 1)) << size
            size += (packet_type_id_enum__step_tmp6 - 1)
            assert isinstance((packet_type_id_enum__value_int_tmp1 != 0), bool)
            result |= ((int((packet_type_id_enum__value_int_tmp1 != 0))) & (2 ** (1) - 1)) << size
            size += 1
            if packet_type_id_enum__value_int_tmp1 == 0: break
        assert isinstance(data.is_valid, bool)
        result |= ((int(data.is_valid)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.period_ago, int)
        assert 0 <= data.period_ago <= 31
        result |= ((data.period_ago) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.value_current, (int, float))
        result |= ((int(round(float(data.value_current) * 100.0, 0)) & 134217727) & (2 ** (27) - 1)) << size
        size += 27
        assert isinstance(data.value_previous_1_delta, (int, float))
        result |= ((int(round(float(data.value_previous_1_delta) * 100.0, 0)) & 16777215) & (2 ** (24) - 1)) << size
        size += 24
        assert isinstance(data.value_previous_2_delta, (int, float))
        result |= ((int(round(float(data.value_previous_2_delta) * 100.0, 0)) & 16777215) & (2 ** (24) - 1)) << size
        size += 24
        assert isinstance(data.value_previous_3_delta, (int, float))
        result |= ((int(round(float(data.value_previous_3_delta) * 100.0, 0)) & 16777215) & (2 ** (24) - 1)) << size
        size += 24
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BRetrospectiveEnergyData':
        result__el_tmp8: typing.Dict[str, typing.Any] = dict()
        packet_type_id_enum__res_tmp9 = 0
        packet_type_id_enum__steps_tmp10 = (8, 3, 3, 3, )
        packet_type_id_enum__res_size_tmp11 = 0
        packet_type_id_enum__step_tmp12 = 0
        for packet_type_id_enum__i_tmp14 in range(32):
            packet_type_id_enum__step_tmp12 = (packet_type_id_enum__steps_tmp10[packet_type_id_enum__i_tmp14] if packet_type_id_enum__i_tmp14 < len(packet_type_id_enum__steps_tmp10) else packet_type_id_enum__steps_tmp10[-1]) - 1
            packet_type_id_enum__res_tmp9 |= buf.shift(packet_type_id_enum__step_tmp12) << packet_type_id_enum__res_size_tmp11
            packet_type_id_enum__res_size_tmp11 += packet_type_id_enum__step_tmp12
            packet_type_id_enum__dff_tmp13 = bool(buf.shift(1))
            if not packet_type_id_enum__dff_tmp13: break
        packet_type_id_enum__buf_tmp15 = buf
        buf = BufRef(packet_type_id_enum__res_tmp9, stop_on_buffer_end=True)
        result__el_tmp8["packet_type_id_enum"] = SmpmUlDeviceEnergy16bRetrospectiveEnergyIds(buf.shift(11))
        buf = packet_type_id_enum__buf_tmp15
        result__el_tmp8["is_valid"] = bool(buf.shift(1))
        result__el_tmp8["period_ago"] = buf.shift(5) + 0
        result__el_tmp8["value_current"] = round(buf.shift(27) / 100.0, 2)
        result__el_tmp8["value_previous_1_delta"] = round(buf.shift(24) / 100.0, 2)
        result__el_tmp8["value_previous_2_delta"] = round(buf.shift(24) / 100.0, 2)
        result__el_tmp8["value_previous_3_delta"] = round(buf.shift(24) / 100.0, 2)
        result = SmpmUlDeviceEnergy16BRetrospectiveEnergyData(**result__el_tmp8)
```

```
        buf.shift(12)
        return result
```

Последнее обновление: 2024-10-24

[Счетчики Электроэнергии] Пакет событий

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_journal`

Описание

Пакет событий счетчика за определенный день.

Каждый раз отправляясь, сообщение дублирует себя же на другом радио-канале.

Канал выбирается случайно (с единственным условием неравенства предыдущему)

Сообщение может быть отправлено строго после smpm_ul_device_energy_16b_daily согласно выбранному временному слоту

Сообщение также может быть запрошено по DownLink

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	8 бит	const(dff(8,3,3,3...))	-	-	-	115 (*hex* **73**)
days_ago	показание на конкретные сутки (дней назад)	7 бит	timedelta	0 секунд	127 дней	-	-
valid	Данные валидны	1 бит	bool	0	1	-	-
time_offset	Позиция времени в дне (от его начала)	6 бит	timedelta	0 секунд	21 час	-	-
journal	-	104 бит	tuple[struct[u13];8]	-	-	-	-
↳ **event**	-	13 бит	struct[u13]	-	-	-	-
↳ **offset**	дельта (количество минут)	5 бит	timedelta	0 секунд	31 минута	-	-
↳ **code**	код события. 0 - означает отсутствие события	8 бит	SmpmUlDeviceEnergy16bJournalType	SmpmUlDeviceEnergy16bJournalType.NONE	SmpmUlDeviceEnergy16bJournalType.SIGNALIZATION_RELAY_STATE_CHANGE	-	-
RESERVED	-	2 бит	-	-	-	-	

Референсные значения типов данных

days_ago

Количество дней назад. 0 - начало сегодняшнего дня.
(на момент отправки 2022-11-22 17:11:12. 3 дня назад означает = начало этого дня = 2022-11-19 00:00:00)

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `127 дней`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`

- Размер: `7 бит`
- тип данных: `timedelta`

valid

Флаг валидности сбрасывается, когда счетчик обнаруживает сброс времени за указанный день (время событий, которые хронологически произошли позже, оказывается меньше времени событий, которые хронологически произошли раньше за этот день). В такой ситуации восстановить точное время событий невозможно.

Данные валидны - Размер: `1 бит`

- тип данных: `bool`

time_offset

Позиция времени в дне (от его начала).
Например сейчас 17:35 => 1055min / 20 => 52 (+ 15min для offset первого события)

Позиция времени в дне (от его начала) - Гранулярность данных: `20 минут`

- Минимальное значение: `0 секунд`
- Максимальное значение: `21 час`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `timedelta`

journal

- Количество элементов: `8`
- Размер: `104 бит`
- тип данных: `tuple[struct[u13];8]`

» event

- Размер: `13 бит`
- тип данных: `struct[u13]`

» OFFSET

Дельта (количество минут).
Время в минутах после предыдущего события в списке (или от начала позиции во времени, для первого события)

дельта (количество минут) - Гранулярность данных: `1 минута`

- Минимальное значение: `0 секунд`
- Максимальное значение: `31 минута`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`

- Размер: `5 бит`
- тип данных: `timedelta`

`» CODE`

код события. 0 - означает отсутствие события

- Минимальное значение: `SmpmUlDeviceEnergy16bJournalType.NONE`
- Максимальное значение: `SmpmUlDeviceEnergy16bJournalType.SIGNALIZATION_RELAY_STATE_CHANGE`
- Размер: `8 бит`
- тип данных: `SmpmUlDeviceEnergy16bJournalType`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	NONE
1	SUCCESSFUL_AUTO_DIAGNOSTIC
2	SWITCH_WINTER_DAYLIGHT
3	SWITCH_SUMMER_DAYLIGHT
4	-- Зарезервировано --
5	SETUP_UPDATE
6	RECORD_DATETIME
7	CHANGE_OFFSET_DAILY_CLOCK
8	PERMISSION_SWITCH_DAYLIGHT_ON
9	PERMISSION_SWITCH_DAYLIGHT_OFF
10	CHANGE_DATE_TIME_SWITCH_DAYLIGHT
11	-- Зарезервировано --
12	ERASE_EEPROM
13	NULLIFY_TARIFF_ACCUMULATION
14	NULLIFY_INTERVAL_ACCUMULATION
15	RESET_PASSWORD
16	RESET_POWER_LOST_TIME_COUNTER
17	RESET_MAGNET_IMPACT_TIME_COUNTER
18	RESET_POWER_INCREASE_TIME_COUNTER
19	RESET_POWER_DECREASE_TIME_COUNTER
20	RESET_MAINTS_FREQ_DIVERGENCE_TIME_COUNTER
21	-- Зарезервировано --
22	RESET_POWER_OVER_LIMIT_TIME_COUNTER
23	-- Зарезервировано --
24	-- Зарезервировано --
25	CHANGE_CAPACITY_DATA_LCD
26	CHANGE_TARIFF_METHODS
27	CHANGE_TARIFF_PROGRAMS
28	CHANGE_ACTUAL_SEASON_SCHEDULES
29	-- Зарезервировано --

Значение	Имя
30	-- Зарезервировано --
31	-- Зарезервировано --
32	-- Зарезервировано --
33	-- Зарезервировано --
34	CHANGE_CONSUMPTION_LIMIT
35	CHANGE_LOW_THRESHOLD_VOLTAGE
36	CHANGE_HIGH_THRESHOLD_VOLTAGE
37	CHANGE_MAINTS_FREQ_THRESHOLD
38	-- Зарезервировано --
39	CHANGE_THRESHOLD_LOW_CONSUMPTION
40	RECHARGE_ENERGY_PAYMENT
41	-- Зарезервировано --
42	-- Зарезервировано --
43	-- Зарезервировано --
44	-- Зарезервировано --
45	-- Зарезервировано --
46	-- Зарезервировано --
47	-- Зарезервировано --
48	-- Зарезервировано --
49	-- Зарезервировано --
50	-- Зарезервировано --
51	-- Зарезервировано --
52	-- Зарезервировано --
53	-- Зарезервировано --
54	-- Зарезервировано --
55	-- Зарезервировано --
56	-- Зарезервировано --
57	-- Зарезервировано --
58	UNSUCCESSFUL_AUTO_DIAGNOSTIC_INTERNAL_CLOCK
59	ABNORMAL_COUNTER_AUTOSTART
60	EXTERNAL_POWER_LOST
61	EXTERNAL_POWER_DETECTED
62	-- Зарезервировано --
63	-- Зарезервировано --
64	-- Зарезервировано --
65	-- Зарезервировано --
66	-- Зарезервировано --
67	-- Зарезервировано --
68	START_POWER_OVER_LIMIT
69	STOP_POWER_OVER_LIMIT
70	ENERGY_OVER_LIMIT_1
71	ENERGY_OVER_LIMIT_2

Значение	Имя
72	ENERGY_OVER_LIMIT_3
73	WRONG_PASSWORD_BLOCK
74	WRONG_PASSWORD_APPEAL
75	EXHAUST_DAILY_BATTERY_LIFE_LIMIT
76	START_MAGNET_IMPACT
77	STOP_MAGNET_IMPACT
78	VIOLATION_TERMINAL_BLOCK_SEAL
79	RECOVERY_TERMINAL_BLOCK_SEAL
80	VIOLATION_CASE_SEAL
81	RECOVERY_CASE_SEAL
82	-- Зарезервировано --
83	-- Зарезервировано --
84	TIME_OUT_SYNC_LIMIT
85	CRITICAL_DIVERGENCE_TIME
86	-- Зарезервировано --
87	-- Зарезервировано --
88	-- Зарезервировано --
89	-- Зарезервировано --
90	OVERHEAT_COUNTER_START
91	OVERHEAT_COUNTER_STOP
92	UNSUCCESSFUL_AUTO_DIAGNOSTIC_MEMORY
93	-- Зарезервировано --
94	LOW_BATTERY_CAPACITY
95	RECOVERY_BATTERY_WORKING_VOLTAGE
96	LOW_CONSUMPTION
97	RESET_FLAG_LOW_CONSUMPTION
98	-- Зарезервировано --
99	-- Зарезервировано --
100	-- Зарезервировано --
101	-- Зарезервировано --
102	-- Зарезервировано --
103	-- Зарезервировано --
104	-- Зарезервировано --
105	-- Зарезервировано --
106	-- Зарезервировано --
107	-- Зарезервировано --
108	-- Зарезервировано --
109	-- Зарезервировано --
110	-- Зарезервировано --
111	-- Зарезервировано --
112	-- Зарезервировано --
113	CHANGE_VALIDATION_SETTINGS

Значение	Имя
114	-- Зарезервировано --
115	-- Зарезервировано --
116	UNSUCCESSFUL_AUTO_DIAGNOSTIC_MEASUREMENT_BLOCK
117	UNSUCCESSFUL_AUTO_DIAGNOSTIC_CALCULATION_BLOCK
118	UNSUCCESSFUL_AUTO_DIAGNOSTIC_POWER_BLOCK
119	UNSUCCESSFUL_AUTO_DIAGNOSTIC_SCREEN
120	UNSUCCESSFUL_AUTO_DIAGNOSTIC_RADIO
121	-- Зарезервировано --
122	-- Зарезервировано --
123	-- Зарезервировано --
124	-- Зарезервировано --
125	-- Зарезервировано --
126	-- Зарезервировано --
127	-- Зарезервировано --
128	-- Зарезервировано --
129	-- Зарезервировано --
130	-- Зарезервировано --
131	-- Зарезервировано --
132	-- Зарезервировано --
133	-- Зарезервировано --
134	MAINS_VOLTAGE_LOST_PHASE_A_START
135	MAINS_VOLTAGE_LOST_PHASE_A_STOP
136	MAINS_VOLTAGE_LOST_PHASE_B_START
137	MAINS_VOLTAGE_LOST_PHASE_B_STOP
138	MAINS_VOLTAGE_LOST_PHASE_C_START
139	MAINS_VOLTAGE_LOST_PHASE_C_STOP
140	VOLTAGE_LAYDOWN_PHASE_A_START
141	VOLTAGE_LAYDOWN_PHASE_A_STOP
142	VOLTAGE_LAYDOWN_PHASE_B_START
143	VOLTAGE_LAYDOWN_PHASE_B_STOP
144	VOLTAGE_LAYDOWN_PHASE_C_START
145	VOLTAGE_LAYDOWN_PHASE_C_STOP
146	OVERVOLTAGE_PHASE_A_START
147	OVERVOLTAGE_PHASE_A_STOP
148	OVERVOLTAGE_PHASE_B_START
149	OVERVOLTAGE_PHASE_B_STOP
150	OVERVOLTAGE_PHASE_C_START
151	OVERVOLTAGE_PHASE_C_STOP
152	OVERCURRENT_PHASE_A_START
153	OVERCURRENT_PHASE_A_STOP
154	OVERCURRENT_PHASE_B_START
155	OVERCURRENT_PHASE_B_STOP

Значение	Имя
156	OVERCURRENT_PHASE_C_START
157	OVERCURRENT_PHASE_C_STOP
158	CURRENT_SUM_THRESHOLD_LOW_START
159	CURRENT_SUM_THRESHOLD_LOW_STOP
160	FREQ_OUT_PHASE_A_START
161	FREQ_OUT_PHASE_A_STOP
162	FREQ_OUT_PHASE_B_START
163	FREQ_OUT_PHASE_B_STOP
164	FREQ_OUT_PHASE_C_START
165	FREQ_OUT_PHASE_C_STOP
166	PHASE_ORDER_DISTURBANCE_START
167	PHASE_ORDER_DISTURBANCE_STOP
168	-- Зарезервировано --
169	RADIO_IMPACT_START
170	RADIO_IMPACT_STOP
171	-- Зарезервировано --
172	-- Зарезервировано --
173	DAYLIGHT_TIME_SWITCH
174	DAYLIGHT_TIME_MODE_DATES_CHANGE
175	INTERNAL_CLOCK_SYNC
176	METROLOGY_CHANGE
177	PROFILE_CONF_CHANGE
178	TARIFFICATION_METHOD_CHANGE
179	PERMISSION_CHANGE_SETTINGS_POWER_CONTROL
180	CONTROL_LEVEL_MAINS_CHANGE
181	PERMISSION_CHANGE_SETTINGS_CONSUMPTION_CONTROL
182	LOAD_RELAY_CONDITION_SETTINGS_CHANGE
183	SIGNALIZATION_RELAY_CONDITION_SETTINGS_CHANGE
184	INTERFACE_SIGNALIZATION_CONDITION_SETTINGS_CHANGE
185	INDICATION_SETTINGS_CHANGE
186	SOUND_SIGNAL_CONDITION_SETTINGS_CHANGE
187	LOAD_RELAY_STATE_CHANGE
188	SIGNALIZATION_RELAY_STATE_CHANGE
189	-- Зарезервировано --
190	-- Зарезервировано --
191	-- Зарезервировано --
192	-- Зарезервировано --
193	-- Зарезервировано --
194	-- Зарезервировано --
195	-- Зарезервировано --
196	-- Зарезервировано --
197	-- Зарезервировано --

Значение	Имя
198	-- Зарезервировано --
199	-- Зарезервировано --
200	-- Зарезервировано --
201	-- Зарезервировано --
202	-- Зарезервировано --
203	-- Зарезервировано --
204	-- Зарезервировано --
205	-- Зарезервировано --
206	-- Зарезервировано --
207	-- Зарезервировано --
208	-- Зарезервировано --
209	-- Зарезервировано --
210	-- Зарезервировано --
211	-- Зарезервировано --
212	-- Зарезервировано --
213	-- Зарезервировано --
214	-- Зарезервировано --
215	-- Зарезервировано --
216	-- Зарезервировано --
217	-- Зарезервировано --
218	-- Зарезервировано --
219	-- Зарезервировано --
220	-- Зарезервировано --
221	-- Зарезервировано --
222	-- Зарезервировано --
223	-- Зарезервировано --
224	-- Зарезервировано --
225	-- Зарезервировано --
226	-- Зарезервировано --
227	-- Зарезервировано --
228	-- Зарезервировано --
229	-- Зарезервировано --
230	-- Зарезервировано --
231	-- Зарезервировано --
232	-- Зарезервировано --
233	-- Зарезервировано --
234	-- Зарезервировано --
235	-- Зарезервировано --
236	-- Зарезервировано --
237	-- Зарезервировано --
238	-- Зарезервировано --
239	-- Зарезервировано --

Значение	Имя
240	-- Зарезервировано --
241	-- Зарезервировано --
242	-- Зарезервировано --
243	-- Зарезервировано --
244	-- Зарезервировано --
245	-- Зарезервировано --
246	-- Зарезервировано --
247	-- Зарезервировано --
248	-- Зарезервировано --
249	-- Зарезервировано --
250	-- Зарезервировано --
251	-- Зарезервировано --
252	-- Зарезервировано --
253	-- Зарезервировано --
254	-- Зарезервировано --
255	-- Зарезервировано --

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__JOURNAL(115)
days_ago: 0.0s
valid: false
time_offset: 14400.0s
journal:

  - offset: 180.0s
    code: SmpmUlDeviceEnergy16bJournalType.CHANGE_OFFSET_DAILY_CLOCK(7)
  - offset: 0.0s
    code: SmpmUlDeviceEnergy16bJournalType.NONE(0)
  - offset: 0.0s
    code: SmpmUlDeviceEnergy16bJournalType.NONE(0)
  - offset: 0.0s
    code: SmpmUlDeviceEnergy16bJournalType.NONE(0)
  - offset: 0.0s
    code: SmpmUlDeviceEnergy16bJournalType.NONE(0)
  - offset: 0.0s
    code: SmpmUlDeviceEnergy16bJournalType.NONE(0)
  - offset: 0.0s
    code: SmpmUlDeviceEnergy16bJournalType.NONE(0)
  - offset: 0.0s
    code: SmpmUlDeviceEnergy16bJournalType.NONE(0)
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	115	1110011
packet_type_id.0.DFF	bool	1	0	0
days_ago	timedelta	7	0	0000000
valid	bool	1	0	0

name	type	size	value(int)	data(bits)
time_offset	timedelta	6	12	001100
journal.0.event.offset	timedelta	5	3	00011
journal.0.event.code	SmpmUlDeviceEnergy16bJournalType	8	7	00000111
journal.1.event.offset	timedelta	5	0	00000
journal.1.event.code	SmpmUlDeviceEnergy16bJournalType	8	0	00000000
journal.2.event.offset	timedelta	5	0	00000
journal.2.event.code	SmpmUlDeviceEnergy16bJournalType	8	0	00000000
journal.3.event.offset	timedelta	5	0	00000
journal.3.event.code	SmpmUlDeviceEnergy16bJournalType	8	0	00000000
journal.4.event.offset	timedelta	5	0	00000
journal.4.event.code	SmpmUlDeviceEnergy16bJournalType	8	0	00000000
journal.5.event.offset	timedelta	5	0	00000
journal.5.event.code	SmpmUlDeviceEnergy16bJournalType	8	0	00000000
journal.6.event.offset	timedelta	5	0	00000
journal.6.event.code	SmpmUlDeviceEnergy16bJournalType	8	0	00000000
journal.7.event.offset	timedelta	5	0	00000
journal.7.event.code	SmpmUlDeviceEnergy16bJournalType	8	0	00000000
RESERVED	u2	2	0	00

Result BIN (MSB - LSB):

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00111000 11001100 00000000 01110011
```

Result HEX (little-endian):

```
73 00 cc 38 00 00 00 00 00 00 00 00 00 00 00 00
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
from enum import IntEnum, unique
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_journal
#
# RESULT int:       952893555
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00111000 11001100 00000000 01110011   LSB
# RESULT hex:  LE    73 00 cc 38 00 00 00 00 00 00 00 00 00 00 00 00
#
# name                  type                           size  value(int)  data(bits)
# --------------------  -----------------------------  ----  ----------  ----------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7                               7         115                                                                                                        1110011
# packet_type_id.0.DFF  bool                             1           0                                                                                                      0
# days_ago              timedelta                        7           0                                                                                                0000000
```

```python
# valid                   bool                                 1           0                                                                                                                  0
# time_offset             timedelta                            6          12                                                                                                            001100
# journal.0.event.offset  timedelta                            5           3                                                                                                       00011
# journal.0.event.code    SmpmUlDeviceEnergy16bJournalType     8           7                                                                                               00000111
# journal.1.event.offset  timedelta                            5           0                                                                                          00000
# journal.1.event.code    SmpmUlDeviceEnergy16bJournalType     8           0                                                                                  00000000
# journal.2.event.offset  timedelta                            5           0                                                                             00000
# journal.2.event.code    SmpmUlDeviceEnergy16bJournalType     8           0                                                                     00000000
# journal.3.event.offset  timedelta                            5           0                                                                00000
# journal.3.event.code    SmpmUlDeviceEnergy16bJournalType     8           0                                                        00000000
# journal.4.event.offset  timedelta                            5           0                                                   00000
# journal.4.event.code    SmpmUlDeviceEnergy16bJournalType     8           0                                           00000000
# journal.5.event.offset  timedelta                            5           0                                      00000
# journal.5.event.code    SmpmUlDeviceEnergy16bJournalType     8           0                              00000000
# journal.6.event.offset  timedelta                            5           0                         00000
# journal.6.event.code    SmpmUlDeviceEnergy16bJournalType     8           0                 00000000
# journal.7.event.offset  timedelta                            5           0            00000
# journal.7.event.code    SmpmUlDeviceEnergy16bJournalType     8           0    00000000
# RESERVED                u2                                   2           0  00
@unique
class SmpmUlDeviceEnergy16bJournalType(IntEnum):
    NONE = 0
    SUCCESSFUL_AUTO_DIAGNOSTIC = 1
    SWITCH_WINTER_DAYLIGHT = 2
    SWITCH_SUMMER_DAYLIGHT = 3
    SETUP_UPDATE = 5
    RECORD_DATETIME = 6
    CHANGE_OFFSET_DAILY_CLOCK = 7
    PERMISSION_SWITCH_DAYLIGHT_ON = 8
    PERMISSION_SWITCH_DAYLIGHT_OFF = 9
    CHANGE_DATE_TIME_SWITCH_DAYLIGHT = 10
    ERASE_EEPROM = 12
    NULLIFY_TARIFF_ACCUMULATION = 13
    NULLIFY_INTERVAL_ACCUMULATION = 14
    RESET_PASSWORD = 15
    RESET_POWER_LOST_TIME_COUNTER = 16
    RESET_MAGNET_IMPACT_TIME_COUNTER = 17
    RESET_POWER_INCREASE_TIME_COUNTER = 18
    RESET_POWER_DECREASE_TIME_COUNTER = 19
    RESET_MAINTS_FREQ_DIVERGENCE_TIME_COUNTER = 20
    RESET_POWER_OVER_LIMIT_TIME_COUNTER = 22
    CHANGE_CAPACITY_DATA_LCD = 25
    CHANGE_TARIFF_METHODS = 26
    CHANGE_TARIFF_PROGRAMS = 27
    CHANGE_ACTUAL_SEASON_SCHEDULES = 28
    CHANGE_CONSUMPTION_LIMIT = 34
    CHANGE_LOW_THRESHOLD_VOLTAGE = 35
    CHANGE_HIGH_THRESHOLD_VOLTAGE = 36
    CHANGE_MAINTS_FREQ_THRESHOLD = 37
    CHANGE_THRESHOLD_LOW_CONSUMPTION = 39
    RECHARGE_ENERGY_PAYMENT = 40
    UNSUCCESSFUL_AUTO_DIAGNOSTIC_INTERNAL_CLOCK = 58
    ABNORMAL_COUNTER_AUTOSTART = 59
    EXTERNAL_POWER_LOST = 60
    EXTERNAL_POWER_DETECTED = 61
    START_POWER_OVER_LIMIT = 68
    STOP_POWER_OVER_LIMIT = 69
    ENERGY_OVER_LIMIT_1 = 70
    ENERGY_OVER_LIMIT_2 = 71
    ENERGY_OVER_LIMIT_3 = 72
    WRONG_PASSWORD_BLOCK = 73
    WRONG_PASSWORD_APPEAL = 74
    EXHAUST_DAILY_BATTERY_LIFE_LIMIT = 75
    START_MAGNET_IMPACT = 76
    STOP_MAGNET_IMPACT = 77
    VIOLATION_TERMINAL_BLOCK_SEAL = 78
    RECOVERY_TERMINAL_BLOCK_SEAL = 79
```

```
    VIOLATION_CASE_SEAL = 80
    RECOVERY_CASE_SEAL = 81
    TIME_OUT_SYNC_LIMIT = 84
    CRITICAL_DIVERGENCE_TIME = 85
    OVERHEAT_COUNTER_START = 90
    OVERHEAT_COUNTER_STOP = 91
    UNSUCCESSFUL_AUTO_DIAGNOSTIC_MEMORY = 92
    LOW_BATTERY_CAPACITY = 94
    RECOVERY_BATTERY_WORKING_VOLTAGE = 95
    LOW_CONSUMPTION = 96
    RESET_FLAG_LOW_CONSUMPTION = 97
    CHANGE_VALIDATION_SETTINGS = 113
    UNSUCCESSFUL_AUTO_DIAGNOSTIC_MEASUREMENT_BLOCK = 116
    UNSUCCESSFUL_AUTO_DIAGNOSTIC_CALCULATION_BLOCK = 117
    UNSUCCESSFUL_AUTO_DIAGNOSTIC_POWER_BLOCK = 118
    UNSUCCESSFUL_AUTO_DIAGNOSTIC_SCREEN = 119
    UNSUCCESSFUL_AUTO_DIAGNOSTIC_RADIO = 120
    MAINS_VOLTAGE_LOST_PHASE_A_START = 134
    MAINS_VOLTAGE_LOST_PHASE_A_STOP = 135
    MAINS_VOLTAGE_LOST_PHASE_B_START = 136
    MAINS_VOLTAGE_LOST_PHASE_B_STOP = 137
    MAINS_VOLTAGE_LOST_PHASE_C_START = 138
    MAINS_VOLTAGE_LOST_PHASE_C_STOP = 139
    VOLTAGE_LAYDOWN_PHASE_A_START = 140
    VOLTAGE_LAYDOWN_PHASE_A_STOP = 141
    VOLTAGE_LAYDOWN_PHASE_B_START = 142
    VOLTAGE_LAYDOWN_PHASE_B_STOP = 143
    VOLTAGE_LAYDOWN_PHASE_C_START = 144
    VOLTAGE_LAYDOWN_PHASE_C_STOP = 145
    OVERVOLTAGE_PHASE_A_START = 146
    OVERVOLTAGE_PHASE_A_STOP = 147
    OVERVOLTAGE_PHASE_B_START = 148
    OVERVOLTAGE_PHASE_B_STOP = 149
    OVERVOLTAGE_PHASE_C_START = 150
    OVERVOLTAGE_PHASE_C_STOP = 151
    OVERCURRENT_PHASE_A_START = 152
    OVERCURRENT_PHASE_A_STOP = 153
    OVERCURRENT_PHASE_B_START = 154
    OVERCURRENT_PHASE_B_STOP = 155
    OVERCURRENT_PHASE_C_START = 156
    OVERCURRENT_PHASE_C_STOP = 157
    CURRENT_SUM_THRESHOLD_LOW_START = 158
    CURRENT_SUM_THRESHOLD_LOW_STOP = 159
    FREQ_OUT_PHASE_A_START = 160
    FREQ_OUT_PHASE_A_STOP = 161
    FREQ_OUT_PHASE_B_START = 162
    FREQ_OUT_PHASE_B_STOP = 163
    FREQ_OUT_PHASE_C_START = 164
    FREQ_OUT_PHASE_C_STOP = 165
    PHASE_ORDER_DISTURBANCE_START = 166
    PHASE_ORDER_DISTURBANCE_STOP = 167
    RADIO_IMPACT_START = 169
    RADIO_IMPACT_STOP = 170
    DAYLIGHT_TIME_SWITCH = 173
    DAYLIGHT_TIME_MODE_DATES_CHANGE = 174
    INTERNAL_CLOCK_SYNC = 175
    METROLOGY_CHANGE = 176
    PROFILE_CONF_CHANGE = 177
    TARIFFICATION_METHOD_CHANGE = 178
    PERMISSION_CHANGE_SETTINGS_POWER_CONTROL = 179
    CONTROL_LEVEL_MAINS_CHANGE = 180
    PERMISSION_CHANGE_SETTINGS_CONSUMPTION_CONTROL = 181
    LOAD_RELAY_CONDITION_SETTINGS_CHANGE = 182
    SIGNALIZATION_RELAY_CONDITION_SETTINGS_CHANGE = 183
    INTERFACE_SIGNALIZATION_CONDITION_SETTINGS_CHANGE = 184
    INDICATION_SETTINGS_CHANGE = 185
    SOUND_SIGNAL_CONDITION_SETTINGS_CHANGE = 186
```

```
    LOAD_RELAY_STATE_CHANGE = 187
    SIGNALIZATION_RELAY_STATE_CHANGE = 188


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class EventData(typing.NamedTuple):
    offset: timedelta
    code: SmpmUlDeviceEnergy16bJournalType
class SmpmUlDeviceEnergy16BJournalData(typing.NamedTuple):
    days_ago: timedelta
    valid: bool
    time_offset: timedelta
    journal: typing.Tuple[EventData, EventData, EventData, EventData, EventData, EventData, EventData, EventData]

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((115) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 127
        result |= ((days_ago_tmp1) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.valid, bool)
        result |= ((int(data.valid)) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.time_offset, (int, timedelta))
        time_offset_tmp2 = int(data.time_offset.total_seconds() // 1200 if isinstance(data.time_offset, timedelta) else data.time_offset // 1200)
        assert 0 <= time_offset_tmp2 <= 63
        result |= ((time_offset_tmp2) & (2 ** (6) - 1)) << size
        size += 6
        assert isinstance(data.journal, tuple) and len(data.journal) == 8
        isinstance(data.journal[0].offset, (int, timedelta))
        offset_tmp3 = int(data.journal[0].offset.total_seconds() // 60 if isinstance(data.journal[0].offset, timedelta) else data.journal[0].offset // 60)
        assert 0 <= offset_tmp3 <= 31
        result |= ((offset_tmp3) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[0].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[0].code.value) & (2 ** (8) - 1)) << size
```

```
        size += 8
        isinstance(data.journal[1].offset, (int, timedelta))
        offset_tmp4 = int(data.journal[1].offset.total_seconds() // 60 if isinstance(data.journal[1].offset, timedelta) else data.journal[1].offset // 60)
        assert 0 <= offset_tmp4 <= 31
        result |= ((offset_tmp4) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[1].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[1].code.value) & (2 ** (8) - 1)) << size
        size += 8
        isinstance(data.journal[2].offset, (int, timedelta))
        offset_tmp5 = int(data.journal[2].offset.total_seconds() // 60 if isinstance(data.journal[2].offset, timedelta) else data.journal[2].offset // 60)
        assert 0 <= offset_tmp5 <= 31
        result |= ((offset_tmp5) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[2].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[2].code.value) & (2 ** (8) - 1)) << size
        size += 8
        isinstance(data.journal[3].offset, (int, timedelta))
        offset_tmp6 = int(data.journal[3].offset.total_seconds() // 60 if isinstance(data.journal[3].offset, timedelta) else data.journal[3].offset // 60)
        assert 0 <= offset_tmp6 <= 31
        result |= ((offset_tmp6) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[3].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[3].code.value) & (2 ** (8) - 1)) << size
        size += 8
        isinstance(data.journal[4].offset, (int, timedelta))
        offset_tmp7 = int(data.journal[4].offset.total_seconds() // 60 if isinstance(data.journal[4].offset, timedelta) else data.journal[4].offset // 60)
        assert 0 <= offset_tmp7 <= 31
        result |= ((offset_tmp7) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[4].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[4].code.value) & (2 ** (8) - 1)) << size
        size += 8
        isinstance(data.journal[5].offset, (int, timedelta))
        offset_tmp8 = int(data.journal[5].offset.total_seconds() // 60 if isinstance(data.journal[5].offset, timedelta) else data.journal[5].offset // 60)
        assert 0 <= offset_tmp8 <= 31
        result |= ((offset_tmp8) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[5].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[5].code.value) & (2 ** (8) - 1)) << size
        size += 8
        isinstance(data.journal[6].offset, (int, timedelta))
        offset_tmp9 = int(data.journal[6].offset.total_seconds() // 60 if isinstance(data.journal[6].offset, timedelta) else data.journal[6].offset // 60)
        assert 0 <= offset_tmp9 <= 31
        result |= ((offset_tmp9) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[6].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[6].code.value) & (2 ** (8) - 1)) << size
        size += 8
        isinstance(data.journal[7].offset, (int, timedelta))
        offset_tmp10 = int(data.journal[7].offset.total_seconds() // 60 if isinstance(data.journal[7].offset, timedelta) else data.journal[7].offset // 60)
        assert 0 <= offset_tmp10 <= 31
        result |= ((offset_tmp10) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.journal[7].code, SmpmUlDeviceEnergy16bJournalType)
        result |= ((data.journal[7].code.value) & (2 ** (8) - 1)) << size
        size += 8
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BJournalData':
        result_el_tmp11: typing.Dict[str, typing.Any] = dict()
        if 115 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result_el_tmp11["days_ago"] = timedelta(seconds=buf.shift(7) * 86400)
```

```
        result__el_tmp11["valid"] = bool(buf.shift(1))
        result__el_tmp11["time_offset"] = timedelta(seconds=buf.shift(6) * 1200)
        journal_tmp12: typing.List[EventData] = []
        journal__item_tmp13__el_tmp14: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp14["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp14["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp14)
        journal_tmp12.append(journal__item_tmp13)
        journal__item_tmp13__el_tmp15: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp15["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp15["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp15)
        journal_tmp12.append(journal__item_tmp13)
        journal__item_tmp13__el_tmp16: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp16["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp16["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp16)
        journal_tmp12.append(journal__item_tmp13)
        journal__item_tmp13__el_tmp17: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp17["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp17["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp17)
        journal_tmp12.append(journal__item_tmp13)
        journal__item_tmp13__el_tmp18: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp18["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp18["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp18)
        journal_tmp12.append(journal__item_tmp13)
        journal__item_tmp13__el_tmp19: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp19["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp19["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp19)
        journal_tmp12.append(journal__item_tmp13)
        journal__item_tmp13__el_tmp20: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp20["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp20["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp20)
        journal_tmp12.append(journal__item_tmp13)
        journal__item_tmp13__el_tmp21: typing.Dict[str, typing.Any] = dict()
        journal__item_tmp13__el_tmp21["offset"] = timedelta(seconds=buf.shift(5) * 60)
        journal__item_tmp13__el_tmp21["code"] = SmpmUlDeviceEnergy16bJournalType(buf.shift(8))
        journal__item_tmp13 = EventData(**journal__item_tmp13__el_tmp21)
        journal_tmp12.append(journal__item_tmp13)
        result__el_tmp11["journal"] = tuple(journal_tmp12)
        result = SmpmUlDeviceEnergy16BJournalData(**result__el_tmp11)
        buf.shift(2)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Пакет показаний потребленной электроэнергии по 3м фазам

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_3phase_consumed`

Описание

Каждый раз отправляясь, сообщение дублирует себя же на другом радио-канале. Канал выбирается случайно (с единственным условием неравенства предыдущему)

Сообщение может быть отправлено строго после smpm_ul_device_energy_16b_daily согласно выбранному временному слоту

Сообщение также может быть запрошено по DownLink

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	716 (*bin* **010 11001100**)
energy_is_reactive	Энергия в пакете Реактивная	1 бит	bool	0	1	-	-
days_ago	показание на конкретные сутки (дней назад)	7 бит	timedelta	0 секунд	127 дней	-	-
valid	Данные валидны	1 бит	bool	0	1	-	-
total	общее значение (в ваттах)	32 бит	u32	0	-	4294967295	-
phase_a	значение на фазе A (в ваттах)	25 бит	u25	0	-	33554431	-
phase_b	значение на фазе B (в ваттах)	25 бит	u25	0	-	33554431	-
phase_c	значение на фазе C (в ваттах)	25 бит	u25	0	-	33554431	-
RESERVED	-	1 бит	-	-	-	-	

Референсные значения типов данных

energy_is_reactive

0 - значит активная энергия
1 - значит реактивная энергия

Энергия в пакете Реактивная - Размер: `1 бит`

- тип данных: `bool`

days_ago

Количество дней назад. 0 - текущие показания на момент отправки сообщения.
Например 3 означает на конец 3 дня от текущего.
(сегодня 2022-11-22 17:11:12. значение должно придти на момент открытия 2022-11-20 00:00:00)

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `127 дней`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `timedelta`

valid

Данные считаются валидными для текущих показаний, если счетчик откалиброван. Данные считаются валидными для архивных показаний (days_ago > 0), если счетчик откалиброван и если данные на указанный период счетчик считает достоверными.

Данные валидны - Размер: `1 бит`

- тип данных: `bool`

total

общее значение (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 4294967295`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `32 бит`
- тип данных: `u32`

phase_a

значение на фазе А (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

phase_b

значение на фазе B (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

phase_c

значение на фазе C (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__ENERGY_CONSUMED_PHASE3(332)
energy_is_reactive: false
days_ago: 0.0s
valid: false
total: 123123
phase_a: 4313
phase_b: 14312123
phase_c: 1312123
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	76	1001100
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	2	10
packet_type_id.1.DFF	bool	1	0	0
energy_is_reactive	bool	1	0	0
days_ago	timedelta	7	0	0000000
valid	bool	1	0	0
total	u32	32	123123	00000000000000011110000011110011

name	type	size	value(int)	data(bits)
phase_a	u25	25	4313	0000000000001000011011001
phase_b	u25	25	14312123	0110110100110001010111011
phase_c	u25	25	1312123	0000101000000010101111011
RESERVED	u1	1	0	0

Result BIN (MSB - LSB):

`00000101 00000001 01011110 11011011 01001100 01010111 01100000 00000001 00001101 10010000 00000000 00011110 00001111 00110000 00000010 11001100`

Result HEX (little-endian):

```
cc 02 30 0f 1e 00 90 0d 01 60 57 4c db 5e 01 05
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_3phase_consumed
#
# RESULT int:        6653256196879938732392428594653037260
# RESULT bin:  MSB   00000101 00000001 01011110 11011011 01001100 01010111 01100000 00000001 00001101 10010000 00000000 00011110 00001111 00110000 00000010 11001100   LSB
# RESULT hex:  LE    cc 02 30 0f 1e 00 90 0d 01 60 57 4c db 5e 01 05
#
# name                  type       size  value(int)                                                                                                                        data(bits)
# --------------------  ---------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7            7          76                                                                                                                           1001100
# packet_type_id.0.DFF  bool          1           1                                                                                                                          1
# packet_type_id.1.VAL  u2            2           2                                                                                                                        10
# packet_type_id.1.DFF  bool          1           0                                                                                                                       0
# energy_is_reactive    bool          1           0                                                                                                                      0
# days_ago              timedelta     7           0                                                                                                               0000000
# valid                 bool          1           0                                                                                                              0
# total                 u32          32      123123                                                                              00000000000000011110000011110011
# phase_a               u25          25        4313                                                     0000000000001000011011001
# phase_b               u25          25    14312123                            0110110100110001010111011
# phase_c               u25          25     1312123   0000101000000010101111011
# RESERVED              u1            1           0  0


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
```

```
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16B3PhaseConsumedData(typing.NamedTuple):
    energy_is_reactive: bool
    days_ago: timedelta
    valid: bool
    total: int
    phase_a: int
    phase_b: int
    phase_c: int

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((76) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((2) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.energy_is_reactive, bool)
        result |= ((int(data.energy_is_reactive)) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 127
        result |= ((days_ago_tmp1) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.valid, bool)
        result |= ((int(data.valid)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.total, int)
        result |= (((data.total) & 4294967295) & (2 ** (32) - 1)) << size
        size += 32
        assert isinstance(data.phase_a, int)
        result |= (((data.phase_a) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.phase_b, int)
        result |= (((data.phase_b) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.phase_c, int)
        result |= (((data.phase_c) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16B3PhaseConsumedData':
        result__el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 76 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 2 != buf.shift(2):
```

```
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["energy_is_reactive"] = bool(buf.shift(1))
        result__el_tmp2["days_ago"] = timedelta(seconds=buf.shift(7) * 86400)
        result__el_tmp2["valid"] = bool(buf.shift(1))
        result__el_tmp2["total"] = buf.shift(32) + 0
        result__el_tmp2["phase_a"] = buf.shift(25) + 0
        result__el_tmp2["phase_b"] = buf.shift(25) + 0
        result__el_tmp2["phase_c"] = buf.shift(25) + 0
        result = SmpmUlDeviceEnergy16B3PhaseConsumedData(**result__el_tmp2)
        buf.shift(1)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Пакет показаний сгенерированной электроэнергии по 3м фазам

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_3phase_generated`

Описание

Каждый раз отправляясь, сообщение дублирует себя же на другом радио-канале. Канал выбирается случайно (с единственным условием неравенства предыдущему)

Сообщение может быть отправлено строго после smpm_ul_device_energy_16b_daily согласно выбранному временному слоту

Сообщение также может быть запрошено по DownLink

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	715 (*bin* **010 11001011**)
energy_is_reactive	Энергия в пакете Реактивная	1 бит	bool	0	1	-	-
days_ago	показание на конкретные сутки (дней назад)	7 бит	timedelta	0 секунд	127 дней	-	-
valid	Данные валидны	1 бит	bool	0	1	-	-
total	общее значение (в ваттах)	32 бит	u32	0	-	4294967295	-
phase_a	значение на фазе A (в ваттах)	25 бит	u25	0	-	33554431	-
phase_b	значение на фазе B (в ваттах)	25 бит	u25	0	-	33554431	-
phase_c	значение на фазе C (в ваттах)	25 бит	u25	0	-	33554431	-
RESERVED	-	1 бит	-	-	-	-	

Референсные значения типов данных

energy_is_reactive

0 - значит активная энергия
1 - значит реактивная энергия

Энергия в пакете Реактивная - Размер: `1 бит`

- тип данных: `bool`

days_ago

Количество дней назад. 0 - текущие показания на момент отправки сообщения.
Например 3 означает на конец 3 дня от текущего.
(сегодня 2022-11-22 17:11:12. значение должно придти на момент открытия 2022-11-20 00:00:00)

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `127 дней`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `timedelta`

valid

Данные считаются валидными для текущих показаний, если счетчик откалиброван. Данные считаются валидными для архивных показаний (days_ago > 0), если счетчик откалиброван и если данные на указанный период счетчик считает достоверными.

Данные валидны - Размер: `1 бит`

- тип данных: `bool`

total

общее значение (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 4294967295`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `32 бит`
- тип данных: `u32`

phase_a

значение на фазе А (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

phase_b

значение на фазе B (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

phase_c

значение на фазе C (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__ENERGY_GENERATED_PHASE3(331)
energy_is_reactive: false
days_ago: 0.0s
valid: false
total: 123123
phase_a: 4313
phase_b: 14312123
phase_c: 1312123
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	75	1001011
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	2	10
packet_type_id.1.DFF	bool	1	0	0
energy_is_reactive	bool	1	0	0
days_ago	timedelta	7	0	0000000
valid	bool	1	0	0
total	u32	32	123123	00000000000000011110000011110011

name	type	size	value(int)	data(bits)
phase_a	u25	25	4313	0000000000001000011011001
phase_b	u25	25	14312123	0110110100110001010111011
phase_c	u25	25	1312123	0000101000000010101111011
RESERVED	u1	1	0	0

Result BIN (MSB - LSB):

`00000101 00000001 01011110 11011011 01001100 01010111 01100000 00000001 00001101 10010000 00000000 00011110 00001111 00110000 00000010 11001011`

Result HEX (little-endian):

```
cb 02 30 0f 1e 00 90 0d 01 60 57 4c db 5e 01 05
```

Парсинг/Сериализация значений

Python   C

```python
from datetime import timedelta
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_3phase_generated
#
# RESULT int:        6653256196879938732392428594653037259
# RESULT bin:  MSB   00000101 00000001 01011110 11011011 01001100 01010111 01100000 00000001 00001101 10010000 00000000 00011110 00001111 00110000 00000010 11001011   LSB
# RESULT hex:  LE    cb 02 30 0f 1e 00 90 0d 01 60 57 4c db 5e 01 05
#
# name                  type       size  value(int)                                                                                                                        data(bits)
# --------------------  ---------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7            7          75                                                                                                                             1001011
# packet_type_id.0.DFF  bool          1           1                                                                                                                            1
# packet_type_id.1.VAL  u2            2           2                                                                                                                          10
# packet_type_id.1.DFF  bool          1           0                                                                                                                         0
# energy_is_reactive    bool          1           0                                                                                                                        0
# days_ago              timedelta     7           0                                                                                                                 0000000
# valid                 bool          1           0                                                                                                                0
# total                 u32          32      123123                                                                                00000000000000011110000011110011
# phase_a               u25          25        4313                                                       0000000000001000011011001
# phase_b               u25          25    14312123                              0110110100110001010111011
# phase_c               u25          25     1312123     0000101000000010101111011
# RESERVED              u1            1           0  0


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
```

```
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16B3PhaseGeneratedData(typing.NamedTuple):
    energy_is_reactive: bool
    days_ago: timedelta
    valid: bool
    total: int
    phase_a: int
    phase_b: int
    phase_c: int

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((75) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((2) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.energy_is_reactive, bool)
        result |= ((int(data.energy_is_reactive)) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 127
        result |= ((days_ago_tmp1) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.valid, bool)
        result |= ((int(data.valid)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.total, int)
        result |= (((data.total) & 4294967295) & (2 ** (32) - 1)) << size
        size += 32
        assert isinstance(data.phase_a, int)
        result |= (((data.phase_a) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.phase_b, int)
        result |= (((data.phase_b) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.phase_c, int)
        result |= (((data.phase_c) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16B3PhaseGeneratedData':
        result_el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 75 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 2 != buf.shift(2):
```

```
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["energy_is_reactive"] = bool(buf.shift(1))
        result__el_tmp2["days_ago"] = timedelta(seconds=buf.shift(7) * 86400)
        result__el_tmp2["valid"] = bool(buf.shift(1))
        result__el_tmp2["total"] = buf.shift(32) + 0
        result__el_tmp2["phase_a"] = buf.shift(25) + 0
        result__el_tmp2["phase_b"] = buf.shift(25) + 0
        result__el_tmp2["phase_c"] = buf.shift(25) + 0
        result = SmpmUlDeviceEnergy16B3PhaseGeneratedData(**result__el_tmp2)
        buf.shift(1)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Пакет общих показаний для счетчиков электроэнергии с маской событий

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_daily`

Описание

Каждый раз отправляясь, сообщение дублирует себя же на другом радио-канале. Канал выбирается случайно (с единственным условием неравенства предыдущему)

Это сообщение является всегда первым для любой серии сообщений. Любая связь с прибором начинается с этого сообщения.

В штатном режиме отправляется трижды в день. Во второй трети дня (08:00 - 16:00) - текущие показания. В первой трети дня (с 00:00 до 08:00) и в третьей трети дня (с 16:00 до 00:00) - показания на предыдущий закрытый день.

Прибор сам определяет во сколько именно ему выйти на связь согласно правилу относительно младшего числа в маке устройства. Например, мак=8912923 число=3. Промежуток выхода на связь будет равен как разделенные 8 часов (480минут) на равные 10 промежутков по 48 минут (начиная с 0). Значит будет выбран по порядку 4й промежуток времени. Например, для первой трети дня это будет (00:00 + 48мин + 48мин + 48мин) = с 01:36 по 02:24 Точное время выбирается случайно согласно полученному промежутку

Сообщение также отправляется каждый раз по наступлению любой из ошибок, и любого воздействия на прибор. В Этом случае отправляются текущие показания

Сообщение также может быть запрошено по DownLink

В случае если часы по какой-то сбросились или отсутствует синхронизация времени дольше чем месяц - прибор переходит в режим отправки сообщений ТОЛЬКО на текущий момент в те же тайм слоты (что предусмотрены для отправки данных на конец суток)

Флаги **ВСЕГДА** соответствуют ТЕКУЩЕМУ состоянию на момент отправки

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	699 (*bin* **010 10111011**)
energy_consumed_active	потребленная энергия активная (в ваттах)	23 бит	u23	0	-	8388607	-
energy_consumed_reactive	потребленная энергия реактивная (в ваттах)	23 бит	u23	0	-	8388607	-
energy_generated_active	сгенерированная энергия активная (в ваттах)	23 бит	u23	0	-	8388607	-
energy_generated_reactive	сгенерированная энергия реактивная (в ваттах)	23 бит	u23	0	-	8388607	-
days_ago	показание на конкретные сутки (дней назад)	7 бит	timedelta	0 секунд	127 дней	-	-
valid	Данные валидны	1 бит	bool	0	1	-	-
error_measurement	Ошибка измерительного блока	1 бит	bool	0	1	-	-
error_low_voltage	Низкий заряд батарейки.Выставляется, когда измеренное значение напряжения батарейки опускается ниже 2.7 В. После выставления, флаг будет снят после восстановления напряжения до 2.9 В и выше.	1 бит	bool	0	1	-	-
error_internal_clock	Ошибка внутренних часов	1 бит	bool	0	1	-	-
error_flash	Ошибка внешней flash памяти	1 бит	bool	0	1	-	-

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
error_eeprom	Ошибка внешней eeprom памяти	1 бит	bool	0	1	-	-
error_radio	Ошибка радио	1 бит	bool	0	1	-	-
error_display	Ошибка индикатора ЖКИ	1 бит	bool	0	1	-	-
error_plc	Ошибка PLC	1 бит	bool	0	1	-	-
error_reset	Внештатный сброс МК	1 бит	bool	0	1	-	-
impact_power_lost	Отключение электричества	1 бит	bool	0	1	-	-
impact_magnet	Магнит активен	1 бит	bool	0	1	-	-
impact_cleat_tamper	Открыт клемник	1 бит	bool	0	1	-	-
impact_body_tamper	Открыт корпус	1 бит	bool	0	1	-	-
impact_radio	Воздействие радио	1 бит	bool	0	1	-	-
RESERVED	-	3 бит	-	-	-	-	

Референсные значения типов данных

energy_consumed_active

потребленная энергия активная (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 8388607`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `23 бит`
- тип данных: `u23`

energy_consumed_reactive

потребленная энергия реактивная (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 8388607`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `23 бит`
- тип данных: `u23`

energy_generated_active

сгенерированная энергия активная (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 8388607` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `23 бит`
- тип данных: `u23`

energy_generated_reactive

сгенерированная энергия реактивная (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 8388607` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `23 бит`
- тип данных: `u23`

days_ago

Количество дней назад. 0 - текущие показания на момент отправки сообщения.
Например 3 означает на конец 3 дня от текущего.
(сегодня 2022-11-22 17:11:12. значение должно придти на момент открытия 2022-11-20 00:00:00)

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `127 дней`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `timedelta`

valid

Данные считаются валидными для текущих показаний, если счетчик откалиброван.
Данные считаются валидными для архивных показаний (days_ago > 0),
если счетчик откалиброван и если данные на указанный период счетчик считает достоверными.

Данные валидны - Размер: `1 бит`

- тип данных: `bool`

error_measurement

Ошибка измерительного блока. Отсутствует обмен между микросхемой измерителя и микроконтроллером счетчика. Ошибка снимается при восстановлении обмена.

Ошибка измерительного блока - Размер: `1 бит`

- тип данных: `bool`

error_low_voltage

Низкий заряд батарейки.Выставляется, когда измеренное значение напряжения батарейки опускается ниже 2.7 В. После выставления, флаг будет снят после восстановления напряжения до 2.9 В и выше. - Размер: `1 бит`

- тип данных: `bool`

error_internal_clock

Ошибка внутренних часов. Выставляется, когда счетчик обнаруживает, что его текущая дата неверна. Активно до тех пор, пока время не станет актуальным (установкой времени, синхронизацией времени).

Ошибка внутренних часов - Размер: `1 бит`

- тип данных: `bool`

error_flash

Ошибка внешней flash памяти. Отсутствует обмен между микросхемой flash памяти и микроконтроллером счетчика. Ошибка выставляется при неудачной попытке доступа счетчика к памяти (запись/чтение). Ошибка снимается при восстановлении обмена.

Ошибка внешней flash памяти - Размер: `1 бит`

- тип данных: `bool`

error_eeprom

Ошибка внешней eeprom памяти. Отсутствует обмен между микросхемой eeprom памяти и микроконтроллером счетчика. Ошибка выставляется при неудачной попытке доступа счетчика к памяти (запись/чтение). Ошибка снимается при восстановлении обмена.

Ошибка внешней eeprom памяти - Размер: `1 бит`

- тип данных: `bool`

error_radio

Ошибка модуля радио. Отсутствует обмен между микросхемой трансивера 433 МГц и микроконтроллером счетчика. Ошибка снимается при восстановлении обмена.

Ошибка радио - Размер: `1 бит`

- тип данных: `bool`

error_display

Ошибка модуля радио. Отсутствует обмен между микросхемой индикатора и микроконтроллером счетчика. Ошибка снимается при восстановлении обмена.

Ошибка индикатора ЖКИ - Размер: `1 бит`

- тип данных: `bool`

error_plc

Ошибка модуля PLC. Выставляется при обнаружении микроконтроллером ошибок при обмене по PLC. Зарезервировано на будущее.

Ошибка PLC - Размер: `1 бит`

- тип данных: `bool`

error_reset

Внештатный автостарт счетчика. Обнаружен сброс микроконтроллера. Флаг активен только при отправке, инициируемой сбросом. В остальные итерации отправки находится в сброшенном состоянии.

Внештатный сброс МК - Размер: `1 бит`

- тип данных: `bool`

impact_power_lost

Флаг указывающий, что было отключение электричества. Присылается в тот момент, когда электричество появилось вновь. В остальные итерации отправки находится в сброшенном состоянии.

Отключение электричества - Размер: `1 бит`

- тип данных: `bool`

impact_magnet

Обнаружено воздействие магнитным полем (любым, в зависимости от установленных датчиков). Сбрасывается при отсутствии воздействия

Магнит активен - Размер: `1 бит`

- тип данных: `bool`

impact_cleat_tamper

Вскрыта клеммная крышка. Выставляется при вскрытии тампера клеммника. Снимается при обратном зажатии тампера.

Открыт клемник - Размер: `1 бит`

- тип данных: `bool`

impact_body_tamper

Вскрыт корпус. Выставляется при вскрытии тампера корпуса.Активно до тех пор, пока сервисная служба не деактивирует этот флаг обратно

Открыт корпус - Размер: `1 бит`

- тип данных: `bool`

impact_radio

Обнаружено воздействие радиополем. Выставляется при обнаружении воздействия, сбрасывается при отсутствии воздействия

Воздействие радио - Размер: `1 бит`

- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__ENERGY(315)
energy_consumed_active: 1430869
energy_consumed_reactive: 1398741
energy_generated_active: 1398101
energy_generated_reactive: 6990523
days_ago: 0.0s
valid: false
error_measurement: true
error_low_voltage: false
error_internal_clock: true
error_flash: true
error_eeprom: false
error_radio: true
error_display: false
error_plc: false
error_reset: false
impact_power_lost: true
impact_magnet: true
impact_cleat_tamper: false
impact_body_tamper: true
impact_radio: false
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	59	0111011
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	2	10
packet_type_id.1.DFF	bool	1	0	0
energy_consumed_active	u23	23	1430869	00101011101010101010101
energy_consumed_reactive	u23	23	1398741	00101010101011111010101
energy_generated_active	u23	23	1398101	00101010101010101010101
energy_generated_reactive	u23	23	6990523	11010101010101010111011
days_ago	timedelta	7	0	0000000
valid	bool	1	0	0
error_measurement	bool	1	1	1
error_low_voltage	bool	1	0	0
error_internal_clock	bool	1	1	1
error_flash	bool	1	1	1
error_eeprom	bool	1	0	0
error_radio	bool	1	1	1
error_display	bool	1	0	0
error_plc	bool	1	0	0

name	type	size	value(int)	data(bits)
error_reset	bool	1	0	0
impact_power_lost	bool	1	1	1
impact_magnet	bool	1	1	1
impact_cleat_tamper	bool	1	0	0
impact_body_tamper	bool	1	1	1
impact_radio	bool	1	0	0
RESERVED	u3	3	0	000

Result BIN (MSB - LSB):

`00001011 00010110 10000000 01101010 10101010 10111011 00101010 10101010 10101010 01010101 01011111 01010100 10101110 10101010 10101010 10111011`

Result HEX (little-endian):

```
bb aa aa ae 54 5f 55 aa aa 2a bb aa 6a 80 16 0b
```

Парсинг/Сериализация значений

Python   C

```python
from datetime import timedelta
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_daily
#
# RESULT int:        14738343083975057021112790317536029371
# RESULT bin:  MSB   00001011 00010110 10000000 01101010 10101010 10111011 00101010 10101010 10101010 01010101 01011111 01010100 10101110 10101010 10101010 10111011   LSB
# RESULT hex:  LE    bb aa aa ae 54 5f 55 aa aa 2a bb aa 6a 80 16 0b
#
# name                       type       size  value(int)                                                                                                                         data(bits)
# ------------------------  ---------  ----  ----------  ----------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL       u7            7          59                                                                                                                            0111011
# packet_type_id.0.DFF       bool          1           1                                                                                                                           1
# packet_type_id.1.VAL       u2            2           2                                                                                                                         10
# packet_type_id.1.DFF       bool          1           0                                                                                                                        0
# energy_consumed_active     u23          23     1430869                                                                                                 00101011101010101010101
# energy_consumed_reactive   u23          23     1398741                                                                          00101010101011111010101
# energy_generated_active    u23          23     1398101                                                   00101010101010101010101
# energy_generated_reactive  u23          23     6990523                            11010101010101010111011
# days_ago                   timedelta     7           0                     0000000
# valid                      bool          1           0                    0
# error_measurement          bool          1           1                   1
# error_low_voltage          bool          1           0                  0
# error_internal_clock       bool          1           1                 1
# error_flash                bool          1           1                1
# error_eeprom               bool          1           0               0
# error_radio                bool          1           1              1
# error_display              bool          1           0             0
# error_plc                  bool          1           0            0
# error_reset                bool          1           0           0
# impact_power_lost          bool          1           1          1
# impact_magnet              bool          1           1         1
```

```
  # impact_cleat_tamper        bool          1           0      0
  # impact_body_tamper         bool          1           1      1
  # impact_radio               bool          1           0     0
  # RESERVED                   u3            3           0  000


  class BufRef:
      def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
          self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
          self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

      def shift(self, size: int) -> int:
          bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
          res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
          buf_len = len(self.value)
          for i in range(bytes_i, needable_len):
              if i >= buf_len:
                  if self._stop_on_buffer_end: break
                  raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
              s = min(8 - bit_i, size - res_size)
              res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
              res_size += s
              bit_i = 0
          self.ends_at += res_size
          return res

      def offset(self, size: int) -> None:
          assert size >= 0, f'{size} was given'
          self.ends_at += size


  class SmpmUlDeviceEnergy16BDailyData(typing.NamedTuple):
      energy_consumed_active: int
      energy_consumed_reactive: int
      energy_generated_active: int
      energy_generated_reactive: int
      days_ago: timedelta
      valid: bool
      error_measurement: bool
      error_low_voltage: bool
      error_internal_clock: bool
      error_flash: bool
      error_eeprom: bool
      error_radio: bool
      error_display: bool
      error_plc: bool
      error_reset: bool
      impact_power_lost: bool
      impact_magnet: bool
      impact_cleat_tamper: bool
      impact_body_tamper: bool
      impact_radio: bool

      def serialize(self) -> bytes:
          data = self
          result = 0
          size = 0
          result |= ((59) & (2 ** (7) - 1)) << size
          size += 7
          result |= ((1) & (2 ** (1) - 1)) << size
          size += 1
          result |= ((2) & (2 ** (2) - 1)) << size
          size += 2
          result |= ((0) & (2 ** (1) - 1)) << size
          size += 1
          assert isinstance(data.energy_consumed_active, int)
          result |= (((data.energy_consumed_active) & 8388607) & (2 ** (23) - 1)) << size
```

```
        size += 23
        assert isinstance(data.energy_consumed_reactive, int)
        result |= (((data.energy_consumed_reactive) & 8388607) & (2 ** (23) - 1)) << size
        size += 23
        assert isinstance(data.energy_generated_active, int)
        result |= (((data.energy_generated_active) & 8388607) & (2 ** (23) - 1)) << size
        size += 23
        assert isinstance(data.energy_generated_reactive, int)
        result |= (((data.energy_generated_reactive) & 8388607) & (2 ** (23) - 1)) << size
        size += 23
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 127
        result |= ((days_ago_tmp1) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.valid, bool)
        result |= ((int(data.valid)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_measurement, bool)
        result |= ((int(data.error_measurement)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_low_voltage, bool)
        result |= ((int(data.error_low_voltage)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_internal_clock, bool)
        result |= ((int(data.error_internal_clock)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_flash, bool)
        result |= ((int(data.error_flash)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_eeprom, bool)
        result |= ((int(data.error_eeprom)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_radio, bool)
        result |= ((int(data.error_radio)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_display, bool)
        result |= ((int(data.error_display)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_plc, bool)
        result |= ((int(data.error_plc)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.error_reset, bool)
        result |= ((int(data.error_reset)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.impact_power_lost, bool)
        result |= ((int(data.impact_power_lost)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.impact_magnet, bool)
        result |= ((int(data.impact_magnet)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.impact_cleat_tamper, bool)
        result |= ((int(data.impact_cleat_tamper)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.impact_body_tamper, bool)
        result |= ((int(data.impact_body_tamper)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.impact_radio, bool)
        result |= ((int(data.impact_radio)) & (2 ** (1) - 1)) << size
        size += 1
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BDailyData':
        result__el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 59 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
```

```
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 2 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["energy_consumed_active"] = buf.shift(23) + 0
        result__el_tmp2["energy_consumed_reactive"] = buf.shift(23) + 0
        result__el_tmp2["energy_generated_active"] = buf.shift(23) + 0
        result__el_tmp2["energy_generated_reactive"] = buf.shift(23) + 0
        result__el_tmp2["days_ago"] = timedelta(seconds=buf.shift(7) * 86400)
        result__el_tmp2["valid"] = bool(buf.shift(1))
        result__el_tmp2["error_measurement"] = bool(buf.shift(1))
        result__el_tmp2["error_low_voltage"] = bool(buf.shift(1))
        result__el_tmp2["error_internal_clock"] = bool(buf.shift(1))
        result__el_tmp2["error_flash"] = bool(buf.shift(1))
        result__el_tmp2["error_eeprom"] = bool(buf.shift(1))
        result__el_tmp2["error_radio"] = bool(buf.shift(1))
        result__el_tmp2["error_display"] = bool(buf.shift(1))
        result__el_tmp2["error_plc"] = bool(buf.shift(1))
        result__el_tmp2["error_reset"] = bool(buf.shift(1))
        result__el_tmp2["impact_power_lost"] = bool(buf.shift(1))
        result__el_tmp2["impact_magnet"] = bool(buf.shift(1))
        result__el_tmp2["impact_cleat_tamper"] = bool(buf.shift(1))
        result__el_tmp2["impact_body_tamper"] = bool(buf.shift(1))
        result__el_tmp2["impact_radio"] = bool(buf.shift(1))
        result = SmpmUlDeviceEnergy16BDailyData(**result__el_tmp2)
        buf.shift(3)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Информационный пакет с параметрами электросети счетчика электроэнергии

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_electricity_params`

Описание

Пакет с мгновенными параметрами электросети

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	956 (*bin* **011 10111100**)
current_ch_1	ток 1-го канала (в амперах)	14 бит	uf14p2	0.0	163.83	-	-
current_ch_2	ток 2-го канала (в амперах)	14 бит	uf14p2	0.0	163.83	-	-
current_ch_3	ток 3-го канала (в амперах)	14 бит	uf14p2	0.0	163.83	-	-
voltage_ch1	напряжение 1-го канала (в вольтах)	9 бит	u9	0	511	-	-
voltage_ch2	напряжение 2-го канала (в вольтах)	9 бит	u9	0	511	-	-
voltage_ch3	напряжение 3-го канала (в вольтах)	9 бит	u9	0	511	-	-
k_ch1	коэффициент мощности 1-го канала (в единицах от -1 до 1)	8 бит	uf8p2	-1.0	1.5499999999999998	-	-
k_ch2	коэффициент мощности 2-го канала (в единицах от -1 до 1)	8 бит	uf8p2	-1.0	1.5499999999999998	-	-
k_ch3	коэффициент мощности 3-го канала(в единицах от -1 до 1)	8 бит	uf8p2	-1.0	1.5499999999999998	-	-
freq	частота в сети (в герцах)	12 бит	uf12p2	30.0	70.95	-	-
RESERVED	-	12 бит	-	-	-	-	

Референсные значения типов данных

current_ch_1

ток 1-го канала (в амперах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `163.83`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`

- Размер: `14 бит`
- тип данных: `uf14p2`

current_ch_2

ток 2-го канала (в амперах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `163.83`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `14 бит`
- тип данных: `uf14p2`

current_ch_3

ток 3-го канала (в амперах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `0.0`
- Максимальное значение: `163.83`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `14 бит`
- тип данных: `uf14p2`

voltage_ch1

напряжение 1-го канала (в вольтах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `511`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `9 бит`
- тип данных: `u9`

voltage_ch2

напряжение 2-го канала (в вольтах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `511`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `9 бит`
- тип данных: `u9`

voltage_ch3

напряжение 3-го канала (в вольтах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `511`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `9 бит`
- тип данных: `u9`

k_ch1

коэффициент мощности 1-го канала (в единицах от -1 до 1)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `-1.0`
- Максимальное значение: `1.5499999999999998`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `8 бит`
- тип данных: `uf8p2`

k_ch2

коэффициент мощности 2-го канала (в единицах от -1 до 1)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `-1.0`
- Максимальное значение: `1.5499999999999998`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `8 бит`
- тип данных: `uf8p2`

k_ch3

коэффициент мощности 3-го канала(в единицах от -1 до 1)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `-1.0`
- Максимальное значение: `1.5499999999999998`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `8 бит`
- тип данных: `uf8p2`

freq

частота в сети (в герцах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `2`
- Минимальное значение: `30.0`
- Максимальное значение: `70.95`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `12 бит`
- тип данных: `uf12p2`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__NETWORK_PARAMS_PHASE1(444)
current_ch_1: 23.3
current_ch_2: 23.3
current_ch_3: 23.3
voltage_ch1: 220
voltage_ch2: 220
voltage_ch3: 220
```

```
k_ch1: 0.33
k_ch2: 0.33
k_ch3: 0.33
freq: 53.3
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	60	0111100
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	3	11
packet_type_id.1.DFF	bool	1	0	0
current_ch_1	uf14p2	14	2330	00100100011010
current_ch_2	uf14p2	14	2330	00100100011010
current_ch_3	uf14p2	14	2330	00100100011010
voltage_ch1	u9	9	220	011011100
voltage_ch2	u9	9	220	011011100
voltage_ch3	u9	9	220	011011100
k_ch1	uf8p2	8	133	10000101
k_ch2	uf8p2	8	133	10000101
k_ch3	uf8p2	8	133	10000101
freq	uf12p2	12	2330	100100011010
RESERVED	u12	12	0	000000000000

Result BIN (MSB - LSB):

`00000000 00001001 00011010 10000101 10000101 10000101 01101110 00110111 00011011 10000100 10001101 00010010 00110100 01001000 11010011 10111100`

Result HEX (little-endian):

```
bc d3 48 34 12 8d 84 1b 37 6e 85 85 85 1a 09 00
```

Парсинг/Сериализация значений

Python C

```python
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_electricity_params
#
# RESULT int:        47268593044936706359241417748435900
# RESULT bin:  MSB   00000000 00001001 00011010 10000101 10000101 10000101 01101110 00110111 00011011 10000100 10001101 00010010 00110100 01001000 11010011 10111100   LSB
# RESULT hex:  LE    bc d3 48 34 12 8d 84 1b 37 6e 85 85 85 1a 09 00
#
# name                  type    size  value(int)                                                                                                                        data(bits)
# --------------------  ------  ----  ----------  ------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7         7          60                                                                                                                           0111100
```

```
# packet_type_id.0.DFF  bool       1          1                                                                                                  1
# packet_type_id.1.VAL  u2         2          3                                                                                                 11
# packet_type_id.1.DFF  bool       1          0                                                                                                 0
# current_ch_1          uf14p2     14      2330                                                                                  00100100011010
# current_ch_2          uf14p2     14      2330                                                                     00100100011010
# current_ch_3          uf14p2     14      2330                                                        00100100011010
# voltage_ch1           u9          9       220                                                 011011100
# voltage_ch2           u9          9       220                                           011011100
# voltage_ch3           u9          9       220                                     011011100
# k_ch1                 uf8p2       8       133                               10000101
# k_ch2                 uf8p2       8       133                         10000101
# k_ch3                 uf8p2       8       133                   10000101
# freq                  uf12p2     12      2330          100100011010
# RESERVED              u12        12         0  000000000000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16BElectricityParamsData(typing.NamedTuple):
    current_ch_1: float
    current_ch_2: float
    current_ch_3: float
    voltage_ch1: int
    voltage_ch2: int
    voltage_ch3: int
    k_ch1: float
    k_ch2: float
    k_ch3: float
    freq: float

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((60) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((3) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.current_ch_1, (int, float))
        assert 0.0 <= data.current_ch_1 <= 163.83
```

```
        result |= ((int(round(float(data.current_ch_1) * 100.0, 0))) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.current_ch_2, (int, float))
        assert 0.0 <= data.current_ch_2 <= 163.83
        result |= ((int(round(float(data.current_ch_2) * 100.0, 0))) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.current_ch_3, (int, float))
        assert 0.0 <= data.current_ch_3 <= 163.83
        result |= ((int(round(float(data.current_ch_3) * 100.0, 0))) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.voltage_ch1, int)
        assert 0 <= data.voltage_ch1 <= 511
        result |= ((data.voltage_ch1) & (2 ** (9) - 1)) << size
        size += 9
        assert isinstance(data.voltage_ch2, int)
        assert 0 <= data.voltage_ch2 <= 511
        result |= ((data.voltage_ch2) & (2 ** (9) - 1)) << size
        size += 9
        assert isinstance(data.voltage_ch3, int)
        assert 0 <= data.voltage_ch3 <= 511
        result |= ((data.voltage_ch3) & (2 ** (9) - 1)) << size
        size += 9
        assert isinstance(data.k_ch1, (int, float))
        assert -1.0 <= data.k_ch1 <= 1.5499999999999998
        result |= ((int(round(float(data.k_ch1 - -1.0) * 100.0, 0))) & (2 ** (8) - 1)) << size
        size += 8
        assert isinstance(data.k_ch2, (int, float))
        assert -1.0 <= data.k_ch2 <= 1.5499999999999998
        result |= ((int(round(float(data.k_ch2 - -1.0) * 100.0, 0))) & (2 ** (8) - 1)) << size
        size += 8
        assert isinstance(data.k_ch3, (int, float))
        assert -1.0 <= data.k_ch3 <= 1.5499999999999998
        result |= ((int(round(float(data.k_ch3 - -1.0) * 100.0, 0))) & (2 ** (8) - 1)) << size
        size += 8
        assert isinstance(data.freq, (int, float))
        assert 30.0 <= data.freq <= 70.95
        result |= ((int(round(float(data.freq - 30.0) * 100.0, 0))) & (2 ** (12) - 1)) << size
        size += 12
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BElectricityParamsData':
        result__el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 60 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 3 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp1["current_ch_1"] = round(buf.shift(14) / 100.0, 2)
        result__el_tmp1["current_ch_2"] = round(buf.shift(14) / 100.0, 2)
        result__el_tmp1["current_ch_3"] = round(buf.shift(14) / 100.0, 2)
        result__el_tmp1["voltage_ch1"] = buf.shift(9) + 0
        result__el_tmp1["voltage_ch2"] = buf.shift(9) + 0
        result__el_tmp1["voltage_ch3"] = buf.shift(9) + 0
        result__el_tmp1["k_ch1"] = round(buf.shift(8) / 100.0 + -1.0, 2)
        result__el_tmp1["k_ch2"] = round(buf.shift(8) / 100.0 + -1.0, 2)
        result__el_tmp1["k_ch3"] = round(buf.shift(8) / 100.0 + -1.0, 2)
        result__el_tmp1["freq"] = round(buf.shift(12) / 100.0 + 30.0, 2)
        result = SmpmUlDeviceEnergy16BElectricityParamsData(**result__el_tmp1)
        buf.shift(12)
        return result
```

[Счетчики Электроэнергии] Информационный пакет счетчика электроэнергии

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_info`

Описание

Каждый раз отправляясь, сообщение дублирует себя же на другом радио-канале. Канал выбирается случайно (с единственным условием неравенства предыдущему)

Сообщение может быть отправлено строго после smpm_ul_device_energy_16b_daily согласно выбранному временному слоту

Сообщение также может быть запрошено по DownLink

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	700 (*bin* **010 10111100**)
battery_volts	уровень заряда батареи (в вольтах)	7 бит	uf7p1	0.0	12.7	-	-
temperature	температура (с точностью до 1 градуса цельсия)	8 бит	u8	-100	155	-	-
datetime	дата в секундах (от 2020-01-01)	31 бит	timedelta	0 секунд	24855 дней 3 часа 14 минуты 7 секунд	-	-
relay_is_active	Активность реле	1 бит	bool	0	1	-	-
RESERVED	-	70 бит	-	-	-	-	

Референсные значения типов данных

battery_volts

уровень заряда батареи (в вольтах)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `1`
- Минимальное значение: `0.0`
- Максимальное значение: `12.7`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `uf7p1`

temperature

температура (с точностью до 1 градуса цельсия)

> Целое Беззнаковое

- Минимальное значение: `-100`
- Максимальное значение: `155`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `8 бит`
- тип данных: `u8`

datetime

дата в секундах (от 2020-01-01)

- Гранулярность данных: `1 секунда`
- Минимальное значение: `0 секунд`
- Максимальное значение: `24855 дней 3 часа 14 минуты 7 секунд`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `31 бит`
- тип данных: `timedelta`

relay_is_active

активность реле. 1 - активно (ток через реле идет). 0 - не активно (ток через реле не идет)

Активность реле - Размер: `1 бит`

- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__INFO(316)
battery_volts: 3.3
temperature: 23
datetime: 567648000.0s
relay_is_active: false
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	60	0111100
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	2	10

name	type	size	value(int)	data(bits)
packet_type_id.1.DFF	bool	1	0	0
battery_volts	uf7p1	7	33	0100001
temperature	u8	8	123	01111011
datetime	timedelta	31	567648000	0100001110101011001111100000000
relay_is_active	bool	1	0	0
RESERVED	u70	70	0	0000000000000000000000000000000000000000000000000000000000000000000000

Result BIN (MSB - LSB):

`00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 10000111 01010110 01111100 00000001 11101101 00001010 10111100`

Result HEX (little-endian):

```
bc 0a ed 01 7c 56 87 00 00 00 00 00 00 00 00 00
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_info
#
# RESULT int:        38094212464183996
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 10000111 01010110 01111100 00000001 11101101 00001010 10111100   LSB
# RESULT hex:  LE    bc 0a ed 01 7c 56 87 00 00 00 00 00 00 00 00 00
#
# name                  type       size  value(int)                                                                                                         data(bits)
# --------------------  ---------  ----  ----------  ----------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7            7          60                                                                                                            0111100
# packet_type_id.0.DFF  bool          1           1                                                                                                           1
# packet_type_id.1.VAL  u2            2           2                                                                                                         10
# packet_type_id.1.DFF  bool          1           0                                                                                                        0
# battery_volts         uf7p1         7          33                                                                                                 0100001
# temperature           u8            8         123                                                                                         01111011
# datetime              timedelta    31   567648000                                                      0100001110101011001111100000000
# relay_is_active       bool          1           0                                                     0
# RESERVED              u70          70           0  0000000000000000000000000000000000000000000000000000000000000000000000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
```

```
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16BInfoData(typing.NamedTuple):
    battery_volts: float
    temperature: int
    datetime: timedelta
    relay_is_active: bool

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((60) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((2) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.battery_volts, (int, float))
        assert 0.0 <= data.battery_volts <= 12.7
        result |= ((int(round(float(data.battery_volts) * 10.0, 0))) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.temperature, int)
        assert -100 <= data.temperature <= 155
        result |= (((data.temperature + 100)) & (2 ** (8) - 1)) << size
        size += 8
        isinstance(data.datetime, (int, timedelta))
        datetime_tmp1 = int(data.datetime.total_seconds() // 1 if isinstance(data.datetime, timedelta) else data.datetime // 1)
        assert 0 <= datetime_tmp1 <= 2147483647
        result |= ((datetime_tmp1) & (2 ** (31) - 1)) << size
        size += 31
        assert isinstance(data.relay_is_active, bool)
        result |= ((int(data.relay_is_active)) & (2 ** (1) - 1)) << size
        size += 1
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BInfoData':
        result__el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 60 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 2 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["battery_volts"] = round(buf.shift(7) / 10.0, 1)
        result__el_tmp2["temperature"] = buf.shift(8) + -100
        result__el_tmp2["datetime"] = timedelta(seconds=buf.shift(31) * 1)
        result__el_tmp2["relay_is_active"] = bool(buf.shift(1))
        result = SmpmUlDeviceEnergy16BInfoData(**result__el_tmp2)
        buf.shift(70)
        return result
```

[Счетчики Электроэнергии] Пакет дневных показаний профилей показаний электросчетчика по часам (00:00-08:00)

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_profile_8h1_energy`

Описание

Данные в пакете профиля рассчитывается на последний законченный час на момент отправки

Для энергий A-, R-, A+, R+

Множитель, умноженный на значение в конкретной точке, равно количество ватт

Точки профиля соответствуют временным интервалам:

1. 00:00 - 01:00
2. 01:00 - 02:00
3. 02:00 - 03:00
4. 03:00 - 04:00
5. 04:00 - 05:00
6. 05:00 - 06:00
7. 06:00 - 07:00
8. 07:00 - 08:00

Для показаний за последние 24 часа отправляются только полностью закрытые профили (отправляются сегодняшние показания, если время >= 08.00; отправляются показания за прошлые сутки, если время < 08.00).

Алгоритм расчета профиля представлен тут

Алгоритм расчета профиля представлен тут

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	8 бит	const(dff(8,3,3,3...))	-	-	-	106 (*hex* **6a**)
type	-	2 бит	SmpmUlDeviceEnergy16bProfile8h1EnergyType	SmpmUlDeviceEnergy16bProfile8h1EnergyType.ENERGY_GENERATED_ACTIVE	SmpmUlDeviceEnergy16bProfile8h1EnergyType.ENERGY_CONSUMED_REACTIVE	-	-
point_factor_multiplier	множитель шага максимума. увеличивает доступный диапозон величин 1..4Выставляется > 1, когда значения в профилях не помещаются в стандартный диапазон (0 - 51603 Вт).	2 бит	u2	1	4	-	-
days_ago	показание на конкретные сутки (дней назад)	6 бит	timedelta	0 секунд	63 дня	-	-
profile	-	104 бит	tuple[u13;8]	-	-	-	-
↳ **point**	Значение профиля	13 бит	u13	0	8191	-	-

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
point_factor	размер шага профиля (с учетом множителя для прибора).	6 бит	uf6p1	0.0	6.3	-	-
RESERVED	-	0 бит	-	-	-	-	

Референсные значения типов данных

type

- Минимальное значение: `SmpmUlDeviceEnergy16bProfile8h1EnergyType.ENERGY_GENERATED_ACTIVE`
- Максимальное значение: `SmpmUlDeviceEnergy16bProfile8h1EnergyType.ENERGY_CONSUMED_REACTIVE`
- Размер: `2 бит`
- тип данных: `SmpmUlDeviceEnergy16bProfile8h1EnergyType`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	ENERGY_GENERATED_ACTIVE
1	ENERGY_GENERATED_REACTIVE
2	ENERGY_CONSUMED_ACTIVE
3	ENERGY_CONSUMED_REACTIVE

point_factor_multiplier

множитель шага максимума. увеличивает доступный диапозон величин 1..4Выставляется > 1, когда значения в профилях не помещаются в стандартный диапазон (0 - 51603 Вт).

> Целое Беззнаковое

- Минимальное значение: `1`
- Максимальное значение: `4`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `2 бит`
- тип данных: `u2`

days_ago

Количество дней назад (либо последние закрытые 24часа).0 - показания за последние 24 ч.Примеры:Текущее время 11.05.2023 10.24.Отправляем профили за 11.05.2023 00-08Текущее время 11.05.2023 18.24.Отправляем профили за 11.05.2023 00-081 и более - кол-во дней назад.Примеры для days_ago=1:Текущее время 11.05.2023 02.24.Отправляем профили за 10.05.2023 00-08Текущее время 11.05.2023 10.24.Отправляем профили за 10.05.2023 00-08Текущее время 11.05.2023 18.24.Отправляем профили за 10.05.2023 00-08

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `63 дня`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `timedelta`

profile

- Количество элементов: `8`
- Размер: `104 бит`
- тип данных: `tuple[u13;8]`

» point

Параметр point_factor у множенный на значение в конкретной точке равен количество ватт в конкретной точке профиля. значение округляется математически (0.5 -> 1)

Значение профиля

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `8191`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `13 бит`
- тип данных: `u13`

point_factor

размер шага профиля (с учетом множителя для прибора).

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `1`
- Минимальное значение: `0.0`
- Максимальное значение: `6.3`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `uf6p1`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__PROFILE_ENERGY__00_TO_08(106)
type: SmpmUlDeviceEnergy16bProfile8h1EnergyType.ENERGY_CONSUMED_ACTIVE(2)
point_factor_multiplier: 2
days_ago: 0.0s
profile:

  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
point_factor: 5.0
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	106	1101010
packet_type_id.0.DFF	bool	1	0	0
type	SmpmUlDeviceEnergy16bProfile8h1EnergyType	2	2	10
point_factor_multiplier	u2	2	1	01
days_ago	timedelta	6	0	000000
profile.0.point	u13	13	0	0000000000000

name	type	size	value(int)	data(bits)
profile.1.point	u13	13	1	0000000000001
profile.2.point	u13	13	2	0000000000010
profile.3.point	u13	13	3	0000000000011
profile.4.point	u13	13	4	0000000000100
profile.5.point	u13	13	5	0000000000101
profile.6.point	u13	13	6	0000000000110
profile.7.point	u13	13	7	0000000000111
point_factor	uf6p1	6	50	110010

Result BIN (MSB - LSB):

`11001000 00000000 11100000 00000110 00000000 00101000 00000001 00000000 00000110 00000000 00100000 00000000 10000000 00000000 00000110 01101010`

Result HEX (little-endian):

```
6a 06 00 80 00 20 00 06 00 01 28 00 06 e0 00 c8
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
from enum import IntEnum, unique
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_profile_8h1_energy
#
# RESULT int:        265850142892151729396105850589001287274
# RESULT bin:  MSB   11001000 00000000 11100000 00000110 00000000 00101000 00000001 00000000 00000110 00000000 00100000 00000000 10000000 00000000 00000110 01101010   LSB
# RESULT hex:  LE    6a 06 00 80 00 20 00 06 00 01 28 00 06 e0 00 c8
#
# name                     type                                        size  value(int)                                                                                                                          data(bits)
# -----------------------  -----------------------------------------   ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL     u7                                             7         106                                                                                                                           1101010
# packet_type_id.0.DFF     bool                                           1           0                                                                                                                          0
# type                     SmpmUlDeviceEnergy16bProfile8h1EnergyType      2           2                                                                                                                        10
# point_factor_multiplier  u2                                             2           1                                                                                                                      01
# days_ago                 timedelta                                      6           0                                                                                                                000000
# profile.0.point          u13                                           13           0                                                                                                   0000000000000
# profile.1.point          u13                                           13           1                                                                                      0000000000001
# profile.2.point          u13                                           13           2                                                                         0000000000010
# profile.3.point          u13                                           13           3                                                            0000000000011
# profile.4.point          u13                                           13           4                                               0000000000100
# profile.5.point          u13                                           13           5                                  0000000000101
# profile.6.point          u13                                           13           6                     0000000000110
# profile.7.point          u13                                           13           7        0000000000111
# point_factor             uf6p1                                          6          50  110010
@unique
class SmpmUlDeviceEnergy16bProfile8h1EnergyType(IntEnum):
    ENERGY_GENERATED_ACTIVE = 0
    ENERGY_GENERATED_REACTIVE = 1
    ENERGY_CONSUMED_ACTIVE = 2
    ENERGY_CONSUMED_REACTIVE = 3


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
```

```
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16BProfile8H1EnergyData(typing.NamedTuple):
    type: SmpmUlDeviceEnergy16bProfile8h1EnergyType
    point_factor_multiplier: int
    days_ago: timedelta
    profile: typing.Tuple[int, int, int, int, int, int, int, int]
    point_factor: float

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((106) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.type, SmpmUlDeviceEnergy16bProfile8h1EnergyType)
        result |= ((data.type.value) & (2 ** (2) - 1)) << size
        size += 2
        assert isinstance(data.point_factor_multiplier, int)
        assert 1 <= data.point_factor_multiplier <= 4
        result |= (((data.point_factor_multiplier + -1)) & (2 ** (2) - 1)) << size
        size += 2
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 63
        result |= ((days_ago_tmp1) & (2 ** (6) - 1)) << size
        size += 6
        assert isinstance(data.profile, tuple) and len(data.profile) == 8
        assert isinstance(data.profile[0], int)
        assert 0 <= data.profile[0] <= 8191
        result |= ((data.profile[0]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[1], int)
        assert 0 <= data.profile[1] <= 8191
        result |= ((data.profile[1]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[2], int)
        assert 0 <= data.profile[2] <= 8191
        result |= ((data.profile[2]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[3], int)
        assert 0 <= data.profile[3] <= 8191
        result |= ((data.profile[3]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[4], int)
        assert 0 <= data.profile[4] <= 8191
        result |= ((data.profile[4]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[5], int)
        assert 0 <= data.profile[5] <= 8191
        result |= ((data.profile[5]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[6], int)
        assert 0 <= data.profile[6] <= 8191
        result |= ((data.profile[6]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[7], int)
        assert 0 <= data.profile[7] <= 8191
        result |= ((data.profile[7]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.point_factor, (int, float))
        assert 0.0 <= data.point_factor <= 6.3
        result |= ((int(round(float(data.point_factor) * 10.0, 0))) & (2 ** (6) - 1)) << size
        size += 6
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BProfile8H1EnergyData':
        result__el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 106 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["type"] = SmpmUlDeviceEnergy16bProfile8h1EnergyType(buf.shift(2))
```

```
        result__el_tmp2["point_factor_multiplier"] = buf.shift(2) + 1
        result__el_tmp2["days_ago"] = timedelta(seconds=buf.shift(6) * 86400)
        profile_tmp3: typing.List[int] = []
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        result__el_tmp2["profile"] = tuple(profile_tmp3)
        result__el_tmp2["point_factor"] = round(buf.shift(6) / 10.0, 1)
        result = SmpmUlDeviceEnergy16BProfile8H1EnergyData(**result__el_tmp2)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Пакет дневных показаний профилей показаний электросчетчика по часам (08:00-16:00)

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_profile_8h2_energy`

Описание

Данные в пакете профиля рассчитывается на последний законченный час на момент отправки

Для энергий A-, R-, A+, R+

Множитель, умноженный на значение в конкретной точке, равно количество ватт

Точки профиля соответствуют временным интервалам:

1. 08:00 - 09:00
2. 09:00 - 10:00
3. 10:00 - 11:00
4. 11:00 - 12:00
5. 12:00 - 13:00
6. 13:00 - 14:00
7. 14:00 - 15:00
8. 15:00 - 16:00

Для показаний за последние 24 часа отправляются только полностью закрытые профили (отправляются сегодняшние показания, если время >= 16.00; отправляются показания за прошлые сутки, если время < 16.00).

Алгоритм расчета профиля представлен тут

Алгоритм расчета профиля представлен тут

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	8 бит	const(dff(8,3,3,3...))	-	-	-	107 (*hex* **6b**)
type	-	2 бит	SmpmUlDeviceEnergy16bProfile8h2EnergyType	SmpmUlDeviceEnergy16bProfile8h2EnergyType.ENERGY_GENERATED_ACTIVE	SmpmUlDeviceEnergy16bProfile8h2EnergyType.ENERGY_CONSUMED_REACTIVE	-	-
point_factor_multiplier	множитель шага максимума. увеличивает доступный диапозон величин 1..4 Выставляется > 1, когда значения в профилях не помещаются в стандартный диапазон (0 - 51603 Вт).	2 бит	u2	1	4	-	-
days_ago	показание на конкретные сутки (дней назад)	6 бит	timedelta	0 секунд	63 дня	-	-
profile	-	104 бит	tuple[u13;8]	-	-	-	-
↳ **point**	Значение профиля	13 бит	u13	0	8191	-	-

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
point_factor	размер шага профиля (с учетом множителя для прибора)	6 бит	uf6p1	0.0	6.3	-	-
RESERVED	-	0 бит	-	-	-	-	

Референсные значения типов данных

type

- Минимальное значение: `SmpmUlDeviceEnergy16bProfile8h2EnergyType.ENERGY_GENERATED_ACTIVE`
- Максимальное значение: `SmpmUlDeviceEnergy16bProfile8h2EnergyType.ENERGY_CONSUMED_REACTIVE`
- Размер: `2 бит`
- тип данных: `SmpmUlDeviceEnergy16bProfile8h2EnergyType`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	ENERGY_GENERATED_ACTIVE
1	ENERGY_GENERATED_REACTIVE
2	ENERGY_CONSUMED_ACTIVE
3	ENERGY_CONSUMED_REACTIVE

point_factor_multiplier

множитель шага максимума. увеличивает доступный диапозон величин 1..4 Выставляется > 1, когда значения в профилях не помещаются в стандартный диапазон (0 - 51603 Вт).

> Целое Беззнаковое

- Минимальное значение: `1`
- Максимальное значение: `4`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `2 бит`
- тип данных: `u2`

days_ago

Количество дней назад (либо последние закрытые 24часа).0 - показания за последние 24 ч.Примеры:Текущее время 11.05.2023 18.24.Отправляем профили за 11.05.2023 08-161 и более - кол-во дней назад.Примеры для days_ago=1:Текущее время 11.05.2023 02.24.Отправляем профили за 10.05.2023 08-16Текущее время 11.05.2023 10.24.Отправляем профили за 10.05.2023 08-16Текущее время 11.05.2023 18.24.Отправляем профили за 10.05.2023 08-16

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `63 дня`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `timedelta`

profile

- Количество элементов: `8`

- Размер: `104 бит`
- тип данных: `tuple[u13;8]`

↠ point

Параметр point_factor у множенный на значение в конкретной точке равен количество ватт в конкретной точке профиля

Значение профиля

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `8191`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `13 бит`
- тип данных: `u13`

point_factor

размер шага профиля (с учетом множителя для прибора)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `1`
- Минимальное значение: `0.0`
- Максимальное значение: `6.3`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `uf6p1`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__PROFILE_ENERGY__08_TO_16(107)
type: SmpmUlDeviceEnergy16bProfile8h2EnergyType.ENERGY_CONSUMED_ACTIVE(2)
point_factor_multiplier: 2
days_ago: 0.0s
profile:

  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
point_factor: 5.0
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	107	1101011
packet_type_id.0.DFF	bool	1	0	0
type	SmpmUlDeviceEnergy16bProfile8h2EnergyType	2	2	10
point_factor_multiplier	u2	2	1	01
days_ago	timedelta	6	0	000000
profile.0.point	u13	13	0	0000000000000
profile.1.point	u13	13	1	0000000000001
profile.2.point	u13	13	2	0000000000010

name	type	size	value(int)	data(bits)
profile.3.point	u13	13	3	0000000000011
profile.4.point	u13	13	4	0000000000100
profile.5.point	u13	13	5	0000000000101
profile.6.point	u13	13	6	0000000000110
profile.7.point	u13	13	7	0000000000111
point_factor	uf6p1	6	50	110010

Result BIN (MSB - LSB):

`11001000 00000000 11100000 00000110 00000000 00101000 00000001 00000000 00000110 00000000 00100000 00000000 10000000 00000000 00000110 01101011`

Result HEX (little-endian):

```
6b 06 00 80 00 20 00 06 00 01 28 00 06 e0 00 c8
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
from enum import IntEnum, unique
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_profile_8h2_energy
#
# RESULT int:        265850142892151729396105850589001287275
# RESULT bin:  MSB   11001000 00000000 11100000 00000110 00000000 00101000 00000001 00000000 00000110 00000000 00100000 00000000 10000000 00000000 00000110 01101011   LSB
# RESULT hex:  LE    6b 06 00 80 00 20 00 06 00 01 28 00 06 e0 00 c8
#
# name                    type                                      size  value(int)                                                                                                                        data(bits)
# ----------------------  ----------------------------------------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL    u7                                           7         107                                                                                                                           1101011
# packet_type_id.0.DFF    bool                                         1           0                                                                                                                          0
# type                    SmpmUlDeviceEnergy16bProfile8h2EnergyType    2           2                                                                                                                        10
# point_factor_multiplier u2                                           2           1                                                                                                                      01
# days_ago                timedelta                                    6           0                                                                                                                000000
# profile.0.point         u13                                         13           0                                                                                                 0000000000000
# profile.1.point         u13                                         13           1                                                                                    0000000000001
# profile.2.point         u13                                         13           2                                                                       0000000000010
# profile.3.point         u13                                         13           3                                                          0000000000011
# profile.4.point         u13                                         13           4                                             0000000000100
# profile.5.point         u13                                         13           5                                0000000000101
# profile.6.point         u13                                         13           6                   0000000000110
# profile.7.point         u13                                         13           7      0000000000111
# point_factor            uf6p1                                        6          50  110010
@unique
class SmpmUlDeviceEnergy16bProfile8h2EnergyType(IntEnum):
    ENERGY_GENERATED_ACTIVE = 0
    ENERGY_GENERATED_REACTIVE = 1
    ENERGY_CONSUMED_ACTIVE = 2
    ENERGY_CONSUMED_REACTIVE = 3


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
```

```python
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16BProfile8H2EnergyData(typing.NamedTuple):
    type: SmpmUlDeviceEnergy16bProfile8h2EnergyType
    point_factor_multiplier: int
    days_ago: timedelta
    profile: typing.Tuple[int, int, int, int, int, int, int, int]
    point_factor: float

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((107) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.type, SmpmUlDeviceEnergy16bProfile8h2EnergyType)
        result |= ((data.type.value) & (2 ** (2) - 1)) << size
        size += 2
        assert isinstance(data.point_factor_multiplier, int)
        assert 1 <= data.point_factor_multiplier <= 4
        result |= (((data.point_factor_multiplier + -1)) & (2 ** (2) - 1)) << size
        size += 2
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 63
        result |= ((days_ago_tmp1) & (2 ** (6) - 1)) << size
        size += 6
        assert isinstance(data.profile, tuple) and len(data.profile) == 8
        assert isinstance(data.profile[0], int)
        assert 0 <= data.profile[0] <= 8191
        result |= ((data.profile[0]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[1], int)
        assert 0 <= data.profile[1] <= 8191
        result |= ((data.profile[1]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[2], int)
        assert 0 <= data.profile[2] <= 8191
        result |= ((data.profile[2]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[3], int)
        assert 0 <= data.profile[3] <= 8191
        result |= ((data.profile[3]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[4], int)
        assert 0 <= data.profile[4] <= 8191
        result |= ((data.profile[4]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[5], int)
        assert 0 <= data.profile[5] <= 8191
        result |= ((data.profile[5]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[6], int)
        assert 0 <= data.profile[6] <= 8191
        result |= ((data.profile[6]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[7], int)
        assert 0 <= data.profile[7] <= 8191
        result |= ((data.profile[7]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.point_factor, (int, float))
        assert 0.0 <= data.point_factor <= 6.3
        result |= ((int(round(float(data.point_factor) * 10.0, 0))) & (2 ** (6) - 1)) << size
        size += 6
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BProfile8H2EnergyData':
        result_el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 107 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result_el_tmp2["type"] = SmpmUlDeviceEnergy16bProfile8h2EnergyType(buf.shift(2))
        result_el_tmp2["point_factor_multiplier"] = buf.shift(2) + 1
        result_el_tmp2["days_ago"] = timedelta(seconds=buf.shift(6) * 86400)
        profile_tmp3: typing.List[int] = []
```

```
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        result__el_tmp2["profile"] = tuple(profile_tmp3)
        result__el_tmp2["point_factor"] = round(buf.shift(6) / 10.0, 1)
        result = SmpmUlDeviceEnergy16BProfile8H2EnergyData(**result__el_tmp2)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Пакет дневных показаний профилей показаний электросчетчика по часам (16:00-24:00)

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_profile_8h3_energy`

Описание

Данные в пакете профиля рассчитывается на последний законченный час на момент отправки Для энергий A-, R-, A+, R+ Множитель, умноженный на значение в конкретной точке, равно количество ватт Точки профиля соответствуют временным интервалам:

1. 16:00 - 17:00
2. 17:00 - 18:00
3. 18:00 - 19:00
4. 19:00 - 20:00
5. 20:00 - 21:00
6. 21:00 - 22:00
7. 22:00 - 23:00
8. 23:00 - 00:00

Для показаний за последние 24 часа отправляются только полностью закрытые профили. за текущие сутки этот пакет отправлен не может быть в принципе

Алгоритм расчета профиля представлен тут

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	8 бит	const(dff(8,3,3,3...))	-	-	-	108 (*hex* **6c**)
type	-	2 бит	SmpmUlDeviceEnergy16bProfile8h3EnergyType	SmpmUlDeviceEnergy16bProfile8h3EnergyType.ENERGY_GENERATED_ACTIVE	SmpmUlDeviceEnergy16bProfile8h3EnergyType.ENERGY_CONSUMED_REACTIVE	-	-
point_factor_multiplier	множитель шага максимума. увеличивает доступный диапозон величин 1..4Выставляется > 1, когда значения в профилях не помещаются в стандартный диапазон (0 - 51603 Вт).	2 бит	u2	1	4	-	-
days_ago	показание на конкретные сутки (дней назад)	6 бит	timedelta	0 секунд	63 дня	-	-
profile	-	104 бит	tuple[u13;8]	-	-	-	-
↳ **point**	Значение профиля	13 бит	u13	0	8191	-	-
point_factor	размер шага профиля (с учетом множителя для прибора)	6 бит	uf6p1	0.0	6.3	-	-

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
RESERVED	-	0 бит	-	-	-	-	

Референсные значения типов данных

type

- Минимальное значение: `SmpmUlDeviceEnergy16bProfile8h3EnergyType.ENERGY_GENERATED_ACTIVE`
- Максимальное значение: `SmpmUlDeviceEnergy16bProfile8h3EnergyType.ENERGY_CONSUMED_REACTIVE`
- Размер: `2 бит`
- тип данных: `SmpmUlDeviceEnergy16bProfile8h3EnergyType`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	ENERGY_GENERATED_ACTIVE
1	ENERGY_GENERATED_REACTIVE
2	ENERGY_CONSUMED_ACTIVE
3	ENERGY_CONSUMED_REACTIVE

point_factor_multiplier

множитель шага максимума. увеличивает доступный диапозон величин 1..4Выставляется > 1, когда значения в профилях не помещаются в стандартный диапазон (0 - 51603 Вт).

> Целое Беззнаковое

- Минимальное значение: `1`
- Максимальное значение: `4`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `2 бит`
- тип данных: `u2`

days_ago

Количество дней назад (либо последние закрытые 24часа).1 и более - кол-во дней назад.Примеры для days_ago=1:Текущее время 11.05.2023 02.24.Отправляем профили за 10.05.2023 16-24Текущее время 11.05.2023 10.24.Отправляем профили за 10.05.2023 16-24Текущее время 11.05.2023 18.24.Отправляем профили за 10.05.2023 16-24

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `63 дня`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `timedelta`

profile

- Количество элементов: `8`
- Размер: `104 бит`
- тип данных: `tuple[u13;8]`

» point

Параметр point_factor у множенный на значение в конкретной точке равен количество ватт в конкретной точке профиля

Значение профиля

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `8191`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `13 бит`
- тип данных: `u13`

point_factor

размер шага профиля (с учетом множителя для прибора)

> Дробное десятичное Беззнаковое с фиксированным количеством знаков после запятой

- Количество знаков после запятой: `1`
- Минимальное значение: `0.0`
- Максимальное значение: `6.3`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `uf6p1`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__PROFILE_ENERGY__16_TO_24(108)
type: SmpmUlDeviceEnergy16bProfile8h3EnergyType.ENERGY_CONSUMED_ACTIVE(2)
point_factor_multiplier: 2
days_ago: 0.0s
profile:

  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
point_factor: 5.0
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	108	1101100
packet_type_id.0.DFF	bool	1	0	0
type	SmpmUlDeviceEnergy16bProfile8h3EnergyType	2	2	10
point_factor_multiplier	u2	2	1	01
days_ago	timedelta	6	0	000000
profile.0.point	u13	13	0	0000000000000
profile.1.point	u13	13	1	0000000000001
profile.2.point	u13	13	2	0000000000010
profile.3.point	u13	13	3	0000000000011
profile.4.point	u13	13	4	0000000000100
profile.5.point	u13	13	5	0000000000101
profile.6.point	u13	13	6	0000000000110

name	type	size	value(int)	data(bits)
profile.7.point	u13	13	7	0000000000111
point_factor	uf6p1	6	50	110010

Result BIN (MSB - LSB):

11001000 00000000 11100000 00000110 00000000 00101000 00000001 00000000 00000110 00000000 00100000 00000000 10000000 00000000 00000110 01101100

Result HEX (little-endian):

```
6c 06 00 80 00 20 00 06 00 01 28 00 06 e0 00 c8
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
from enum import IntEnum, unique
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_profile_8h3_energy
#
# RESULT int:        265850142892151729396105850589001287276
# RESULT bin:  MSB   11001000 00000000 11100000 00000110 00000000 00101000 00000001 00000000 00000110 00000000 00100000 00000000 10000000 00000000 00000110 01101100   LSB
# RESULT hex:  LE    6c 06 00 80 00 20 00 06 00 01 28 00 06 e0 00 c8
#
# name                    type                                    size  value(int)  data(bits)
# ----------------------  --------------------------------------  ----  ----------  ----------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL    u7                                         7         108                                                                                                     1101100
# packet_type_id.0.DFF    bool                                       1           0                                                                                                    0
# type                    SmpmUlDeviceEnergy16bProfile8h3EnergyType  2           2                                                                                                   10
# point_factor_multiplier u2                                         2           1                                                                                                01
# days_ago                timedelta                                  6           0                                                                                       000000
# profile.0.point         u13                                       13           0                                                                          0000000000000
# profile.1.point         u13                                       13           1                                                                0000000000001
# profile.2.point         u13                                       13           2                                                    0000000000010
# profile.3.point         u13                                       13           3                                        0000000000011
# profile.4.point         u13                                       13           4                            0000000000100
# profile.5.point         u13                                       13           5                0000000000101
# profile.6.point         u13                                       13           6    0000000000110
# profile.7.point         u13                                       13           7  0000000000111
# point_factor            uf6p1                                      6          50  110010
@unique
class SmpmUlDeviceEnergy16bProfile8h3EnergyType(IntEnum):
    ENERGY_GENERATED_ACTIVE = 0
    ENERGY_GENERATED_REACTIVE = 1
    ENERGY_CONSUMED_ACTIVE = 2
    ENERGY_CONSUMED_REACTIVE = 3


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size
```

```
class SmpmUlDeviceEnergy16BProfile8H3EnergyData(typing.NamedTuple):
    type: SmpmUlDeviceEnergy16bProfile8h3EnergyType
    point_factor_multiplier: int
    days_ago: timedelta
    profile: typing.Tuple[int, int, int, int, int, int, int, int]
    point_factor: float

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((108) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.type, SmpmUlDeviceEnergy16bProfile8h3EnergyType)
        result |= ((data.type.value) & (2 ** (2) - 1)) << size
        size += 2
        assert isinstance(data.point_factor_multiplier, int)
        assert 1 <= data.point_factor_multiplier <= 4
        result |= (((data.point_factor_multiplier + -1)) & (2 ** (2) - 1)) << size
        size += 2
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 63
        result |= ((days_ago_tmp1) & (2 ** (6) - 1)) << size
        size += 6
        assert isinstance(data.profile, tuple) and len(data.profile) == 8
        assert isinstance(data.profile[0], int)
        assert 0 <= data.profile[0] <= 8191
        result |= ((data.profile[0]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[1], int)
        assert 0 <= data.profile[1] <= 8191
        result |= ((data.profile[1]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[2], int)
        assert 0 <= data.profile[2] <= 8191
        result |= ((data.profile[2]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[3], int)
        assert 0 <= data.profile[3] <= 8191
        result |= ((data.profile[3]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[4], int)
        assert 0 <= data.profile[4] <= 8191
        result |= ((data.profile[4]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[5], int)
        assert 0 <= data.profile[5] <= 8191
        result |= ((data.profile[5]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[6], int)
        assert 0 <= data.profile[6] <= 8191
        result |= ((data.profile[6]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.profile[7], int)
        assert 0 <= data.profile[7] <= 8191
        result |= ((data.profile[7]) & (2 ** (13) - 1)) << size
        size += 13
        assert isinstance(data.point_factor, (int, float))
        assert 0.0 <= data.point_factor <= 6.3
        result |= ((int(round(float(data.point_factor) * 10.0, 0))) & (2 ** (6) - 1)) << size
        size += 6
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BProfile8H3EnergyData':
        result__el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 108 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["type"] = SmpmUlDeviceEnergy16bProfile8h3EnergyType(buf.shift(2))
        result__el_tmp2["point_factor_multiplier"] = buf.shift(2) + 1
        result__el_tmp2["days_ago"] = timedelta(seconds=buf.shift(6) * 86400)
        profile_tmp3: typing.List[int] = []
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
```

```
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        profile__item_tmp4 = buf.shift(13) + 0
        profile_tmp3.append(profile__item_tmp4)
        result__el_tmp2["profile"] = tuple(profile_tmp3)
        result__el_tmp2["point_factor"] = round(buf.shift(6) / 10.0, 1)
        result = SmpmUlDeviceEnergy16BProfile8H3EnergyData(**result__el_tmp2)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Пакет показаний тарифов потребления электроэнергии

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_tariff_consumed`

Описание

Каждый раз отправляясь, сообщение дублирует себя же на другом радио-канале. Канал выбирается случайно (с единственным условием неравенства предыдущему)

Сообщение может быть отправлено строго после smpm_ul_device_energy_16b_daily согласно выбранному временному слоту

Сообщение также может быть запрошено по DownLink

Тариф может содержать только 4 тарифа конкретной энергии (только задействованные тарифы включая аварийны тариф). При этом какие из 8 - указаны в маске тарифов. Тарифы распределяются последовательно согласно их порядковому номеру. Если какие-либо тарифы не указаны, то значения в слоте заполняются нулями

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	706 (*bin* **010 11000010**)
energy_is_reactive	Энергия в пакете Реактивная	1 бит	bool	0	1	-	-
days_ago	показание на конкретные сутки (дней назад)	7 бит	timedelta	0 секунд	127 дней	-	-
valid	Данные валидны	1 бит	bool	0	1	-	-
tariff_mask	Маска тарифов в пакете	8 бит	tuple[bool;8]	-	-	-	-
↳ **tariff**	если 1 то тариф под этим номером пришел	1 бит	bool	0	1	-	-
slot_0	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
slot_1	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
slot_2	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
slot_3	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
RESERVED	-	0 бит	-	-	-	-	

Референсные значения типов данных

energy_is_reactive

0 - значит активная энергия
1 - значит реактивная энергия

Энергия в пакете Реактивная - Размер: `1 бит`

- тип данных: `bool`

days_ago

Количество дней назад. 0 - текущие показания на момент отправки сообщения.
Например 3 означает на конец 3 дня от текущего.
(сегодня 2022-11-22 17:11:12. значение должно придти на момент открытия 2022-11-20 00:00:00)

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `127 дней`
- Лимитирование значения: `нет`

- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `timedelta`

valid

Данные считаются валидными для текущих показаний, если счетчик откалиброван. Данные считаются валидными для архивных показаний (days_ago > 0), если счетчик откалиброван и если данные на указанный период счетчик считает достоверными.

Данные валидны - Размер: `1 бит`

- тип данных: `bool`

tariff_mask

Говорит о том какой именно из 8 тарифов пришел.

Позиция единицы означает номер тарифа

Всего в этом пакете можно передать максимальное количество 4 тарифа

Маска тарифов в пакете - Количество элементов: `8` - Размер: `8 бит`

- тип данных: `tuple[bool;8]`

↠ tariff

если 1 то тариф под этим номером пришел

- Размер: `1 бит`
- тип данных: `bool`

slot_0

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

slot_1

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431` . При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

slot_2

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

slot_3

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__ENERGY_CONSUMED_TARIFF(322)
energy_is_reactive: false
days_ago: 0.0s
valid: false
tariff_mask:

  - true
  - false
  - true
  - false
  - false
  - true
  - false
  - false
slot_0: 123123
slot_1: 4312123
slot_2: 5413
slot_3: 0
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	66	1000010
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	2	10
packet_type_id.1.DFF	bool	1	0	0
energy_is_reactive	bool	1	0	0
days_ago	timedelta	7	0	0000000
valid	bool	1	0	0
tariff_mask.0.tariff	bool	1	1	1
tariff_mask.1.tariff	bool	1	0	0
tariff_mask.2.tariff	bool	1	1	1
tariff_mask.3.tariff	bool	1	0	0
tariff_mask.4.tariff	bool	1	0	0
tariff_mask.5.tariff	bool	1	1	1
tariff_mask.6.tariff	bool	1	0	0

name	type	size	value(int)	data(bits)
tariff_mask.7.tariff	bool	1	0	0
slot_0	u25	25	123123	0000000011110000011110011
slot_1	u25	25	4312123	0010000011100110000111011
slot_2	u25	25	5413	0000000000001010100100101
slot_3	u25	25	0	0000000000000000000000000

Result BIN (MSB - LSB):

```
00000000 00000000 00000000 00000000 00000101 01001001 01001000 00111001 10000111 01100000 00011110 00001111 00110010 01010000 00000010 11000010
```

Result HEX (little-endian):

```
c2 02 50 32 0f 1e 60 87 39 48 49 05 00 00 00 00
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_tariff_consumed
#
# RESULT int:        16360177055446019327792225794
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00000101 01001001 01001000 00111001 10000111 01100000 00011110 00001111 00110010 01010000 00000010 11000010   LSB
# RESULT hex:  LE    c2 02 50 32 0f 1e 60 87 39 48 49 05 00 00 00 00
#
# name                  type       size  value(int)                                                                                                                        data(bits)
# --------------------  ---------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7            7          66                                                                                                                           1000010
# packet_type_id.0.DFF  bool          1           1                                                                                                                          1
# packet_type_id.1.VAL  u2            2           2                                                                                                                        10
# packet_type_id.1.DFF  bool          1           0                                                                                                                       0
# energy_is_reactive    bool          1           0                                                                                                                      0
# days_ago              timedelta     7           0                                                                                                               0000000
# valid                 bool          1           0                                                                                                              0
# tariff_mask.0.tariff  bool          1           1                                                                                                             1
# tariff_mask.1.tariff  bool          1           0                                                                                                            0
# tariff_mask.2.tariff  bool          1           1                                                                                                           1
# tariff_mask.3.tariff  bool          1           0                                                                                                          0
# tariff_mask.4.tariff  bool          1           0                                                                                                         0
# tariff_mask.5.tariff  bool          1           1                                                                                                        1
# tariff_mask.6.tariff  bool          1           0                                                                                                       0
# tariff_mask.7.tariff  bool          1           0                                                                                                      0
# slot_0                u25          25      123123                                                                             0000000011110000011110011
# slot_1                u25          25     4312123                                                    0010000011100110000111011
# slot_2                u25          25        5413                           0000000000001010100100101
# slot_3                u25          25           0  0000000000000000000000000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
```

```
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmUlDeviceEnergy16BTariffConsumedData(typing.NamedTuple):
    energy_is_reactive: bool
    days_ago: timedelta
    valid: bool
    tariff_mask: typing.Tuple[bool, bool, bool, bool, bool, bool, bool, bool]
    slot_0: int
    slot_1: int
    slot_2: int
    slot_3: int

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((66) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((2) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.energy_is_reactive, bool)
        result |= ((int(data.energy_is_reactive)) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 127
        result |= ((days_ago_tmp1) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.valid, bool)
        result |= ((int(data.valid)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask, tuple) and len(data.tariff_mask) == 8
        assert isinstance(data.tariff_mask[0], bool)
        result |= ((int(data.tariff_mask[0])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[1], bool)
        result |= ((int(data.tariff_mask[1])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[2], bool)
        result |= ((int(data.tariff_mask[2])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[3], bool)
        result |= ((int(data.tariff_mask[3])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[4], bool)
        result |= ((int(data.tariff_mask[4])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[5], bool)
        result |= ((int(data.tariff_mask[5])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[6], bool)
        result |= ((int(data.tariff_mask[6])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[7], bool)
        result |= ((int(data.tariff_mask[7])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.slot_0, int)
        result |= (((data.slot_0) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.slot_1, int)
        result |= (((data.slot_1) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.slot_2, int)
        result |= (((data.slot_2) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.slot_3, int)
        result |= (((data.slot_3) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BTariffConsumedData':
        result__el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 66 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 2 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
```

```
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["energy_is_reactive"] = bool(buf.shift(1))
        result__el_tmp2["days_ago"] = timedelta(seconds=buf.shift(7) * 86400)
        result__el_tmp2["valid"] = bool(buf.shift(1))
        tariff_mask_tmp3: typing.List[bool] = []
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        result__el_tmp2["tariff_mask"] = tuple(tariff_mask_tmp3)
        result__el_tmp2["slot_0"] = buf.shift(25) + 0
        result__el_tmp2["slot_1"] = buf.shift(25) + 0
        result__el_tmp2["slot_2"] = buf.shift(25) + 0
        result__el_tmp2["slot_3"] = buf.shift(25) + 0
        result = SmpmUlDeviceEnergy16BTariffConsumedData(**result__el_tmp2)
        return result
```

Последнее обновление: 2023-07-13

[Счетчики Электроэнергии] Пакет показаний тарифов сгенерированной электроэнергии

Тип пакета: `Uplink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. UpLink. Пакеты показаний прибора`

Внутреннее имя: `smpm_ul_device_energy_16b_tariff_generated`

Описание

Каждый раз отправляясь, сообщение дублирует себя же на другом радио-канале. Канал выбирается случайно (с единственным условием неравенства предыдущему)

Сообщение может быть отправлено строго после smpm_ul_device_energy_16b_daily согласно выбранному временному слоту

Сообщение также может быть запрошено по DownLink

Тариф может содержать только 4 тарифа конкретной энергии (только задействованные тарифы включая аварийны тариф). При этом какие из 8 - указаны в маске тарифов. Тарифы распределяются последовательно согласно их порядковому номеру. Если какие-либо тарифы не указаны, то значения в слоте заполняются нулями

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	705 (*bin* **010 11000001**)
energy_is_reactive	Энергия в пакете Реактивная	1 бит	bool	0	1	-	-
days_ago	показание на конкретные сутки (дней назад)	7 бит	timedelta	0 секунд	127 дней	-	-
valid	Данные валидны	1 бит	bool	0	1	-	-
tariff_mask	Маска тарифов в пакете	8 бит	tuple[bool;8]	-	-	-	-
↳ **tariff**	если 1 то тариф под этим номером пришел	1 бит	bool	0	1	-	-
slot_0	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
slot_1	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
slot_2	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
slot_3	Слот для значения (в ваттах)	25 бит	u25	0	-	33554431	-
RESERVED	-	0 бит	-	-	-	-	

Референсные значения типов данных

energy_is_reactive

0 - значит активная энергия
1 - значит реактивная энергия

Энергия в пакете Реактивная - Размер: `1 бит`

- тип данных: `bool`

days_ago

Количество дней назад. 0 - текущие показания на момент отправки сообщения.
Например 3 означает на конец 3 дня от текущего.
(сегодня 2022-11-22 17:11:12. значение должно придти на момент открытия 2022-11-20 00:00:00)

показание на конкретные сутки (дней назад) - Гранулярность данных: `1 день`

- Минимальное значение: `0 секунд`
- Максимальное значение: `127 дней`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `timedelta`

valid

Данные считаются валидными для текущих показаний, если счетчик откалиброван. Данные считаются валидными для архивных показаний (days_ago > 0), если счетчик откалиброван и если данные на указанный период счетчик считает достоверными.

Данные валидны - Размер: `1 бит`

- тип данных: `bool`

tariff_mask

Говорит о том какой именно из 8 тарифов пришел.

Позиция единицы означает номер тарифа

Всего в этом пакете можно передать максимальное количество 4 тарифа

Маска тарифов в пакете - Количество элементов: `8` - Размер: `8 бит`

- тип данных: `tuple[bool;8]`

» tariff

если 1 то тариф под этим номером пришел

- Размер: `1 бит`
- тип данных: `bool`

slot_0

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`

- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

slot_1

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

slot_2

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

slot_3

Слот для значения (в ваттах)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `Не определено (так как возможно переполнение)`
- Лимитирование значения: `нет`
- Переполнение значения: `Каждые 33554431`. При этом ожидается что система принимающая пакеты учитывает переполнение сама
- Размер: `25 бит`
- тип данных: `u25`

Пример пакета

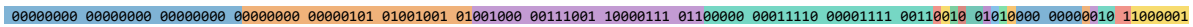
```
packet_type_id: UpLinkDeviceEnergyPackId.UL_DATA_16B__ENERGY_GENERATED_TARIFF(321)
energy_is_reactive: false
days_ago: 0.0s
valid: false
tariff_mask:

  - true
  - false
  - true
  - false
  - false
  - true
  - false
  - false
slot_0: 123123
slot_1: 4312123
slot_2: 5413
slot_3: 0
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	65	1000001
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	2	10
packet_type_id.1.DFF	bool	1	0	0
energy_is_reactive	bool	1	0	0
days_ago	timedelta	7	0	0000000
valid	bool	1	0	0
tariff_mask.0.tariff	bool	1	1	1
tariff_mask.1.tariff	bool	1	0	0
tariff_mask.2.tariff	bool	1	1	1
tariff_mask.3.tariff	bool	1	0	0
tariff_mask.4.tariff	bool	1	0	0
tariff_mask.5.tariff	bool	1	1	1
tariff_mask.6.tariff	bool	1	0	0
tariff_mask.7.tariff	bool	1	0	0
slot_0	u25	25	123123	0000000011110000011110011
slot_1	u25	25	4312123	0010000011100110000111011
slot_2	u25	25	5413	0000000000001010100100101
slot_3	u25	25	0	0000000000000000000000000

Result BIN (MSB - LSB):



Result HEX (little-endian):

```
c1 02 50 32 0f 1e 60 87 39 48 49 05 00 00 00 00
```

Парсинг/Сериализация значений

Python   C

```python
from datetime import timedelta
import math
import typing


# PACKET (128 bits)   smpm_ul_device_energy_16b_tariff_generated
#
# RESULT int:        163601770554460193277922579З
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00000101 01001001 01001000 00111001 10000111 01100000 00011110 00001111 00110010 01010000 00000010 11000001   LSB
# RESULT hex:  LE    c1 02 50 32 0f 1e 60 87 39 48 49 05 00 00 00 00
#
# name                  type       size  value(int)                                                                                                                        data(bits)
# --------------------  ---------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7            7          65                                                                                                                           1000001
# packet_type_id.0.DFF  bool          1           1                                                                                                                          1
# packet_type_id.1.VAL  u2            2           2                                                                                                                        10
# packet_type_id.1.DFF  bool          1           0                                                                                                                       0
# energy_is_reactive    bool          1           0                                                                                                                      0
# days_ago              timedelta     7           0                                                                                                               0000000
# valid                 bool          1           0                                                                                                              0
# tariff_mask.0.tariff  bool          1           1                                                                                                             1
# tariff_mask.1.tariff  bool          1           0                                                                                                            0
# tariff_mask.2.tariff  bool          1           1                                                                                                           1
# tariff_mask.3.tariff  bool          1           0                                                                                                          0
# tariff_mask.4.tariff  bool          1           0                                                                                                         0
# tariff_mask.5.tariff  bool          1           1                                                                                                        1
# tariff_mask.6.tariff  bool          1           0                                                                                                       0
# tariff_mask.7.tariff  bool          1           0                                                                                                      0
# slot_0                u25          25      123123                                                                             0000000011110000011110011
# slot_1                u25          25     4312123                                                    0010000011100110000111011
# slot_2                u25          25        5413                           0000000000001010100100101
# slot_3                u25          25           0  0000000000000000000000000


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size
```

```
class SmpmUlDeviceEnergy16BTariffGeneratedData(typing.NamedTuple):
    energy_is_reactive: bool
    days_ago: timedelta
    valid: bool
    tariff_mask: typing.Tuple[bool, bool, bool, bool, bool, bool, bool, bool]
    slot_0: int
    slot_1: int
    slot_2: int
    slot_3: int

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((65) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((2) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.energy_is_reactive, bool)
        result |= ((int(data.energy_is_reactive)) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.days_ago, (int, timedelta))
        days_ago_tmp1 = int(data.days_ago.total_seconds() // 86400 if isinstance(data.days_ago, timedelta) else data.days_ago // 86400)
        assert 0 <= days_ago_tmp1 <= 127
        result |= ((days_ago_tmp1) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.valid, bool)
        result |= ((int(data.valid)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask, tuple) and len(data.tariff_mask) == 8
        assert isinstance(data.tariff_mask[0], bool)
        result |= ((int(data.tariff_mask[0])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[1], bool)
        result |= ((int(data.tariff_mask[1])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[2], bool)
        result |= ((int(data.tariff_mask[2])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[3], bool)
        result |= ((int(data.tariff_mask[3])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[4], bool)
        result |= ((int(data.tariff_mask[4])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[5], bool)
        result |= ((int(data.tariff_mask[5])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[6], bool)
        result |= ((int(data.tariff_mask[6])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.tariff_mask[7], bool)
        result |= ((int(data.tariff_mask[7])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.slot_0, int)
        result |= (((data.slot_0) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.slot_1, int)
        result |= (((data.slot_1) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.slot_2, int)
```

```
        result |= (((data.slot_2) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        assert isinstance(data.slot_3, int)
        result |= (((data.slot_3) & 33554431) & (2 ** (25) - 1)) << size
        size += 25
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmUlDeviceEnergy16BTariffGeneratedData':
        result__el_tmp2: typing.Dict[str, typing.Any] = dict()
        if 65 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 2 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp2["energy_is_reactive"] = bool(buf.shift(1))
        result__el_tmp2["days_ago"] = timedelta(seconds=buf.shift(7) * 86400)
        result__el_tmp2["valid"] = bool(buf.shift(1))
        tariff_mask_tmp3: typing.List[bool] = []
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        tariff_mask__item_tmp4 = bool(buf.shift(1))
        tariff_mask_tmp3.append(tariff_mask__item_tmp4)
        result__el_tmp2["tariff_mask"] = tuple(tariff_mask_tmp3)
        result__el_tmp2["slot_0"] = buf.shift(25) + 0
        result__el_tmp2["slot_1"] = buf.shift(25) + 0
        result__el_tmp2["slot_2"] = buf.shift(25) + 0
        result__el_tmp2["slot_3"] = buf.shift(25) + 0
        result = SmpmUlDeviceEnergy16BTariffGeneratedData(**result__el_tmp2)
        return result
```

[Счетчики Электроэнергии] Пакет установки регулярной отправки пакетов с данными

Тип пакета: `Downlink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. Downlink. Пакеты показаний прибора`

Внутреннее имя: `smpm_dl_device_energy_16b_set_regular_data_sending`

Описание

Пакет установки расписания отправки пакетов с данными

Маска DAY WEEK MONTH работает как условие, при котором несколько выбранных в каждой маске значений означает как еще одна отправка в рамках Месяцев/ДнейНедели/ДнейМесяца/Часов. Выбрав 7 дней недели - значит все дни недели (каким бы условие дней не было). Выбрав все 31 дня - все дни месяца выбрав все 24 часа - 24 раза в день будет отправка

Но если выбрать только понедельник, 31день, 3месяц. То сработает это расписание, только тогда когда текущий день будет именно 31го марта и когда это будет понедельник

Случаи:

1) Установка пакета каждый день в промежуток с 12:00 до 13:00 - week_mask: 1111111 - days_mask: 11111111111111111111111111111111 - hour_mask: 00000000000100000000000

2) Установка пакета только по понедельникам и воскресеньям 04:00..05.00 и 23:00...00:00 - week_mask: 1000001 - days_mask: 11111111111111111111111111111111 - hour_mask: 00001000000000000000001

3) Отправлять пакет первые 5 дней каждого месяца 12:00 до 13:00 - week_mask: 1111111 - days_mask: 11111000000000000000000000000000 - hour_mask: 00000000000100000000000

4) Выключить отправку пакета - action: REPLACE - week_mask: 0000000 - days_mask: 00000000000000000000000000000000 - hour_mask: 00000000000000000000000

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-
pack_id	ID пакета (число, без учета DFF)	14 бит	SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId	SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId.UL_DATA_16B__ENERGY	SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId.MONTHLY_ENERGY_REACTIVE_GENERATED
action	Действие Замены/ Удаления/ Добавления	2 бит	SpecUpLinkModelingPacketIdU8Action	SpecUpLinkModelingPacketIdU8Action.REPLACE	SpecUpLinkModelingPacketIdU8Action.REPLACE_DAY
period_ago	Маркер периода назад.	6 бит	u6	0	63
week_mask	Маска дней недели отправки	7 бит	tuple[bool;7]	-	-
↳ **day**	-	1 бит	bool	0	1

Поле	Описание	Размер	Тип	Мин	Макс
days_mask	Маска дней в месяце отправки	31 бит	tuple[bool;31]	-	-
↳ day	-	1 бит	bool	0	1
hour_mask	Маска часов отправки	24 бит	tuple[bool;24]	-	-
↳ hour	-	1 бит	bool	0	1
RESERVED	для downlink-пакетов резервное место заполняется рандомными битами	33 бит	-	-	-

Референсные значения типов данных

pack_id

ID пакета (число, без учета DFF)

- Минимальное значение: `SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId.UL_DATA_16B__ENERGY`
- Максимальное значение: `SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId.MONTHLY_ENERGY_REACTIVE_GENERATED`
- Размер: `14 бит`
- тип данных: `SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
315	UL_DATA_16B__ENERGY
400	DAILY_ENERGY_ACTIVE_CONSUMED
401	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1
402	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2
403	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3
404	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4
405	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM
406	DAILY_ENERGY_REACTIVE_CONSUMED
407	DAILY_ENERGY_ACTIVE_GENERATED
408	DAILY_ENERGY_REACTIVE_GENERATED
409	MONTHLY_ENERGY_ACTIVE_CONSUMED
410	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1
411	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2
412	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3
413	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4
414	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM

Значение	Имя
415	MONTHLY_ENERGY_REACTIVE_CONSUMED
416	MONTHLY_ENERGY_ACTIVE_GENERATED
417	MONTHLY_ENERGY_REACTIVE_GENERATED

action

Действие Замены/Удаления/Добавления

- Минимальное значение: `SpecUpLinkModelingPacketIdU8Action.REPLACE`
- Максимальное значение: `SpecUpLinkModelingPacketIdU8Action.REPLACE_DAY`
- Размер: `2 бит`
- тип данных: `SpecUpLinkModelingPacketIdU8Action`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	REPLACE
1	ADD
2	REPLACE_HOUR
3	REPLACE_DAY

period_ago

Маркер периода назад.
(0 = текущий день, текущие показания)
При отправке пакета данные будут браться из архива на конкретный момент в прошлом (относительно момента отправки)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `63`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `6 бит`
- тип данных: `u6`

week_mask

Маска дней недели отправки

- Количество элементов: `7`
- Размер: `7 бит`
- тип данных: `tuple[bool;7]`

» day

- Размер: `1 бит`
- тип данных: `bool`

days_mask

Маска дней в месяце отправки

- Количество элементов: `31`
- Размер: `31 бит`
- тип данных: `tuple[bool;31]`

day

- Размер: `1 бит`
- тип данных: `bool`

hour_mask

Маска часов отправки

- Количество элементов: `24`
- Размер: `24 бит`
- тип данных: `tuple[bool;24]`

hour

- Размер: `1 бит`
- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyDlPackId.SET_REGULAR_DATA_SENDING(150)
pack_id: SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId.UL_DATA_16B__ENERGY(315)
action: SpecUpLinkModelingPacketIdU8Action.REPLACE(0)
period_ago: 31
week_mask:

  - false
  - false
  - false
  - false
  - false
  - false
  - false
days_mask:

  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
```

```
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
hour_mask:

  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
  - false
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	22	0010110
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	1	01
packet_type_id.1.DFF	bool	1	0	0
pack_id	SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId	14	315	00000100111011
action	SpecUpLinkModelingPacketIdU8Action	2	0	00
period_ago	u6	6	31	011111
week_mask.0.day	bool	1	0	0
week_mask.1.day	bool	1	0	0

name	type	size	value(int)	data(bits)
week_mask.2.day	bool	1	0	0
week_mask.3.day	bool	1	0	0
week_mask.4.day	bool	1	0	0
week_mask.5.day	bool	1	0	0
week_mask.6.day	bool	1	0	0
days_mask.0.day	bool	1	0	0
days_mask.1.day	bool	1	0	0
days_mask.2.day	bool	1	0	0
days_mask.3.day	bool	1	0	0
days_mask.4.day	bool	1	0	0
days_mask.5.day	bool	1	0	0
days_mask.6.day	bool	1	0	0
days_mask.7.day	bool	1	0	0
days_mask.8.day	bool	1	0	0
days_mask.9.day	bool	1	0	0
days_mask.10.day	bool	1	0	0
days_mask.11.day	bool	1	0	0
days_mask.12.day	bool	1	0	0
days_mask.13.day	bool	1	0	0
days_mask.14.day	bool	1	0	0
days_mask.15.day	bool	1	0	0
days_mask.16.day	bool	1	0	0
days_mask.17.day	bool	1	0	0
days_mask.18.day	bool	1	0	0
days_mask.19.day	bool	1	0	0
days_mask.20.day	bool	1	0	0
days_mask.21.day	bool	1	0	0
days_mask.22.day	bool	1	0	0
days_mask.23.day	bool	1	0	0
days_mask.24.day	bool	1	0	0
days_mask.25.day	bool	1	0	0
days_mask.26.day	bool	1	0	0
days_mask.27.day	bool	1	0	0
days_mask.28.day	bool	1	0	0
days_mask.29.day	bool	1	0	0
days_mask.30.day	bool	1	0	0
hour_mask.0.hour	bool	1	0	0
hour_mask.1.hour	bool	1	0	0
hour_mask.2.hour	bool	1	0	0
hour_mask.3.hour	bool	1	0	0
hour_mask.4.hour	bool	1	0	0
hour_mask.5.hour	bool	1	0	0
hour_mask.6.hour	bool	1	0	0

name	type	size	value(int)	data(bits)
hour_mask.7.hour	bool	1	0	0
hour_mask.8.hour	bool	1	0	0
hour_mask.9.hour	bool	1	0	0
hour_mask.10.hour	bool	1	0	0
hour_mask.11.hour	bool	1	0	0
hour_mask.12.hour	bool	1	0	0
hour_mask.13.hour	bool	1	0	0
hour_mask.14.hour	bool	1	0	0
hour_mask.15.hour	bool	1	0	0
hour_mask.16.hour	bool	1	0	0
hour_mask.17.hour	bool	1	0	0
hour_mask.18.hour	bool	1	0	0
hour_mask.19.hour	bool	1	0	0
hour_mask.20.hour	bool	1	0	0
hour_mask.21.hour	bool	1	0	0
hour_mask.22.hour	bool	1	0	0
hour_mask.23.hour	bool	1	0	0
RESERVED	u33	33	0	000000000000000000000000000000000

Result BIN (MSB - LSB):

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111000 00001001 11011001 10010110
```

Result HEX (little-endian):

```
96 d9 09 f8 00 00 00 00 00 00 00 00 00 00 00 00
```

Парсинг/Сериализация значений

Python C

```python
from enum import IntEnum, unique
import math
import random
import typing


# PACKET (128 bits)   smpm_dl_device_energy_16b_set_regular_data_sending
#
# RESULT int:        4161395094
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111000 00001001 11011001 10010110   LSB
# RESULT hex:  LE    96 d9 09 f8 00 00 00 00 00 00 00 00 00 00 00 00
#
# name                 type                                                        size  value(int)                                                                                                                        data(bits)
# -------------------  ----------------------------------------------------------  ----  ----------  --------------------------------------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL  u7                                                            7          22                                                                                                                           0010110
# packet_type_id.0.DFF  bool                                                          1           1                                                                                                                          1
# packet_type_id.1.VAL  u2                                                            2           1                                                                                                                        01
# packet_type_id.1.DFF  bool                                                          1           0                                                                                                                       0
# pack_id              SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId     14         315                                                                                                         00000100111011
```

```
# action              SpecUpLinkModelingPacketIdU8Action    2       0                                                                                                       00
# period_ago          u6                                    6      31                                                                                                 011111
# week_mask.0.day     bool                                  1       0                                                                                                0
# week_mask.1.day     bool                                  1       0                                                                                               0
# week_mask.2.day     bool                                  1       0                                                                                              0
# week_mask.3.day     bool                                  1       0                                                                                             0
# week_mask.4.day     bool                                  1       0                                                                                            0
# week_mask.5.day     bool                                  1       0                                                                                           0
# week_mask.6.day     bool                                  1       0                                                                                          0
# days_mask.0.day     bool                                  1       0                                                                                         0
# days_mask.1.day     bool                                  1       0                                                                                        0
# days_mask.2.day     bool                                  1       0                                                                                       0
# days_mask.3.day     bool                                  1       0                                                                                      0
# days_mask.4.day     bool                                  1       0                                                                                     0
# days_mask.5.day     bool                                  1       0                                                                                    0
# days_mask.6.day     bool                                  1       0                                                                                   0
# days_mask.7.day     bool                                  1       0                                                                                  0
# days_mask.8.day     bool                                  1       0                                                                                 0
# days_mask.9.day     bool                                  1       0                                                                                0
# days_mask.10.day    bool                                  1       0                                                                               0
# days_mask.11.day    bool                                  1       0                                                                              0
# days_mask.12.day    bool                                  1       0                                                                             0
# days_mask.13.day    bool                                  1       0                                                                            0
# days_mask.14.day    bool                                  1       0                                                                           0
# days_mask.15.day    bool                                  1       0                                                                          0
# days_mask.16.day    bool                                  1       0                                                                         0
# days_mask.17.day    bool                                  1       0                                                                        0
# days_mask.18.day    bool                                  1       0                                                                       0
# days_mask.19.day    bool                                  1       0                                                                      0
# days_mask.20.day    bool                                  1       0                                                                     0
# days_mask.21.day    bool                                  1       0                                                                    0
# days_mask.22.day    bool                                  1       0                                                                   0
# days_mask.23.day    bool                                  1       0                                                                  0
# days_mask.24.day    bool                                  1       0                                                                 0
# days_mask.25.day    bool                                  1       0                                                                0
# days_mask.26.day    bool                                  1       0                                                               0
# days_mask.27.day    bool                                  1       0                                                              0
# days_mask.28.day    bool                                  1       0                                                             0
# days_mask.29.day    bool                                  1       0                                                            0
# days_mask.30.day    bool                                  1       0                                                           0
# hour_mask.0.hour    bool                                  1       0                                                          0
# hour_mask.1.hour    bool                                  1       0                                                         0
# hour_mask.2.hour    bool                                  1       0                                                        0
# hour_mask.3.hour    bool                                  1       0                                                       0
# hour_mask.4.hour    bool                                  1       0                                                      0
# hour_mask.5.hour    bool                                  1       0                                                     0
# hour_mask.6.hour    bool                                  1       0                                                    0
# hour_mask.7.hour    bool                                  1       0                                                   0
# hour_mask.8.hour    bool                                  1       0                                                  0
# hour_mask.9.hour    bool                                  1       0                                                 0
# hour_mask.10.hour   bool                                  1       0                                                0
# hour_mask.11.hour   bool                                  1       0                                               0
# hour_mask.12.hour   bool                                  1       0                                              0
# hour_mask.13.hour   bool                                  1       0                                             0
# hour_mask.14.hour   bool                                  1       0                                            0
# hour_mask.15.hour   bool                                  1       0                                           0
# hour_mask.16.hour   bool                                  1       0                                          0
# hour_mask.17.hour   bool                                  1       0                                         0
# hour_mask.18.hour   bool                                  1       0                                        0
# hour_mask.19.hour   bool                                  1       0                                       0
# hour_mask.20.hour   bool                                  1       0                                      0
# hour_mask.21.hour   bool                                  1       0                                     0
# hour_mask.22.hour   bool                                  1       0                                    0
# hour_mask.23.hour   bool                                  1       0                                   0
# RESERVED            u33                                  33       0  000000000000000000000000000000000
# Reserved space is filled with random bits in DOWNLINK packets !
@unique
class SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId(IntEnum):
    UL_DATA_16B__ENERGY = 315
```

```
    DAILY_ENERGY_ACTIVE_CONSUMED = 400
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 401
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 402
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 403
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 404
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 405
    DAILY_ENERGY_REACTIVE_CONSUMED = 406
    DAILY_ENERGY_ACTIVE_GENERATED = 407
    DAILY_ENERGY_REACTIVE_GENERATED = 408
    MONTHLY_ENERGY_ACTIVE_CONSUMED = 409
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 410
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 411
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 412
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 413
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 414
    MONTHLY_ENERGY_REACTIVE_CONSUMED = 415
    MONTHLY_ENERGY_ACTIVE_GENERATED = 416
    MONTHLY_ENERGY_REACTIVE_GENERATED = 417



@unique
class SpecUpLinkModelingPacketIdU8Action(IntEnum):
    REPLACE = 0
    ADD = 1
    REPLACE_HOUR = 2
    REPLACE_DAY = 3


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmDlDeviceEnergy16BSetRegularDataSendingData(typing.NamedTuple):
    pack_id: SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId
    action: SpecUpLinkModelingPacketIdU8Action
    period_ago: int
    week_mask: typing.Tuple[bool, bool, bool, bool, bool, bool, bool]
    days_mask: typing.Tuple[bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool]
    hour_mask: typing.Tuple[bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool, bool]

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((22) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
```

```
        size += 1
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.pack_id, SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId)
        result |= ((data.pack_id.value) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.action, SpecUpLinkModelingPacketIdU8Action)
        result |= ((data.action.value) & (2 ** (2) - 1)) << size
        size += 2
        assert isinstance(data.period_ago, int)
        assert 0 <= data.period_ago <= 63
        result |= ((data.period_ago) & (2 ** (6) - 1)) << size
        size += 6
        assert isinstance(data.week_mask, tuple) and len(data.week_mask) == 7
        assert isinstance(data.week_mask[0], bool)
        result |= ((int(data.week_mask[0])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.week_mask[1], bool)
        result |= ((int(data.week_mask[1])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.week_mask[2], bool)
        result |= ((int(data.week_mask[2])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.week_mask[3], bool)
        result |= ((int(data.week_mask[3])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.week_mask[4], bool)
        result |= ((int(data.week_mask[4])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.week_mask[5], bool)
        result |= ((int(data.week_mask[5])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.week_mask[6], bool)
        result |= ((int(data.week_mask[6])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask, tuple) and len(data.days_mask) == 31
        assert isinstance(data.days_mask[0], bool)
        result |= ((int(data.days_mask[0])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[1], bool)
        result |= ((int(data.days_mask[1])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[2], bool)
        result |= ((int(data.days_mask[2])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[3], bool)
        result |= ((int(data.days_mask[3])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[4], bool)
        result |= ((int(data.days_mask[4])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[5], bool)
        result |= ((int(data.days_mask[5])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[6], bool)
        result |= ((int(data.days_mask[6])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[7], bool)
        result |= ((int(data.days_mask[7])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[8], bool)
        result |= ((int(data.days_mask[8])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[9], bool)
        result |= ((int(data.days_mask[9])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[10], bool)
```

```
        result |= ((int(data.days_mask[10])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[11], bool)
        result |= ((int(data.days_mask[11])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[12], bool)
        result |= ((int(data.days_mask[12])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[13], bool)
        result |= ((int(data.days_mask[13])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[14], bool)
        result |= ((int(data.days_mask[14])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[15], bool)
        result |= ((int(data.days_mask[15])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[16], bool)
        result |= ((int(data.days_mask[16])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[17], bool)
        result |= ((int(data.days_mask[17])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[18], bool)
        result |= ((int(data.days_mask[18])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[19], bool)
        result |= ((int(data.days_mask[19])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[20], bool)
        result |= ((int(data.days_mask[20])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[21], bool)
        result |= ((int(data.days_mask[21])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[22], bool)
        result |= ((int(data.days_mask[22])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[23], bool)
        result |= ((int(data.days_mask[23])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[24], bool)
        result |= ((int(data.days_mask[24])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[25], bool)
        result |= ((int(data.days_mask[25])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[26], bool)
        result |= ((int(data.days_mask[26])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[27], bool)
        result |= ((int(data.days_mask[27])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[28], bool)
        result |= ((int(data.days_mask[28])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[29], bool)
        result |= ((int(data.days_mask[29])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.days_mask[30], bool)
        result |= ((int(data.days_mask[30])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask, tuple) and len(data.hour_mask) == 24
        assert isinstance(data.hour_mask[0], bool)
        result |= ((int(data.hour_mask[0])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[1], bool)
        result |= ((int(data.hour_mask[1])) & (2 ** (1) - 1)) << size
        size += 1
```

```
        assert isinstance(data.hour_mask[2], bool)
        result |= ((int(data.hour_mask[2])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[3], bool)
        result |= ((int(data.hour_mask[3])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[4], bool)
        result |= ((int(data.hour_mask[4])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[5], bool)
        result |= ((int(data.hour_mask[5])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[6], bool)
        result |= ((int(data.hour_mask[6])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[7], bool)
        result |= ((int(data.hour_mask[7])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[8], bool)
        result |= ((int(data.hour_mask[8])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[9], bool)
        result |= ((int(data.hour_mask[9])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[10], bool)
        result |= ((int(data.hour_mask[10])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[11], bool)
        result |= ((int(data.hour_mask[11])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[12], bool)
        result |= ((int(data.hour_mask[12])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[13], bool)
        result |= ((int(data.hour_mask[13])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[14], bool)
        result |= ((int(data.hour_mask[14])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[15], bool)
        result |= ((int(data.hour_mask[15])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[16], bool)
        result |= ((int(data.hour_mask[16])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[17], bool)
        result |= ((int(data.hour_mask[17])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[18], bool)
        result |= ((int(data.hour_mask[18])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[19], bool)
        result |= ((int(data.hour_mask[19])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[20], bool)
        result |= ((int(data.hour_mask[20])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[21], bool)
        result |= ((int(data.hour_mask[21])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[22], bool)
        result |= ((int(data.hour_mask[22])) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.hour_mask[23], bool)
        result |= ((int(data.hour_mask[23])) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((random.getrandbits(33)) & (2 ** 33 - 1)) << size
        size += 33
        return result.to_bytes(16, "little")
```

```python
@classmethod
def parse(cls, buf: BufRef) -> 'SmpmDlDeviceEnergy16BSetRegularDataSendingData':
    result__el_tmp1: typing.Dict[str, typing.Any] = dict()
    if 22 != buf.shift(7):
        raise ValueError("packet_type_id: buffer doesn't match value")
    if 1 != buf.shift(1):
        raise ValueError("packet_type_id: buffer doesn't match value")
    if 1 != buf.shift(2):
        raise ValueError("packet_type_id: buffer doesn't match value")
    if 0 != buf.shift(1):
        raise ValueError("packet_type_id: buffer doesn't match value")
    result__el_tmp1["pack_id"] = SmpmDlDeviceEnergy16bSetRegularDataSendingDataRequestId(buf.shift(14))
    result__el_tmp1["action"] = SpecUpLinkModelingPacketIdU8Action(buf.shift(2))
    result__el_tmp1["period_ago"] = buf.shift(6) + 0
    week_mask_tmp2: typing.List[bool] = []
    week_mask__item_tmp3 = bool(buf.shift(1))
    week_mask_tmp2.append(week_mask__item_tmp3)
    week_mask__item_tmp3 = bool(buf.shift(1))
    week_mask_tmp2.append(week_mask__item_tmp3)
    week_mask__item_tmp3 = bool(buf.shift(1))
    week_mask_tmp2.append(week_mask__item_tmp3)
    week_mask__item_tmp3 = bool(buf.shift(1))
    week_mask_tmp2.append(week_mask__item_tmp3)
    week_mask__item_tmp3 = bool(buf.shift(1))
    week_mask_tmp2.append(week_mask__item_tmp3)
    week_mask__item_tmp3 = bool(buf.shift(1))
    week_mask_tmp2.append(week_mask__item_tmp3)
    week_mask__item_tmp3 = bool(buf.shift(1))
    week_mask_tmp2.append(week_mask__item_tmp3)
    result__el_tmp1["week_mask"] = tuple(week_mask_tmp2)
    days_mask_tmp4: typing.List[bool] = []
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
    days_mask_tmp4.append(days_mask__item_tmp5)
    days_mask__item_tmp5 = bool(buf.shift(1))
```

```
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        days_mask__item_tmp5 = bool(buf.shift(1))
        days_mask_tmp4.append(days_mask__item_tmp5)
        result__el_tmp1["days_mask"] = tuple(days_mask_tmp4)
        hour_mask_tmp6: typing.List[bool] = []
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
```

```
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        hour_mask__item_tmp7 = bool(buf.shift(1))
        hour_mask_tmp6.append(hour_mask__item_tmp7)
        result__el_tmp1["hour_mask"] = tuple(hour_mask_tmp6)
        result = SmpmDlDeviceEnergy16BSetRegularDataSendingData(**result__el_tmp1)
        buf.shift(33)
        return result
```

Последнее обновление: 2024-10-24

[Счетчики Электроэнергии] Пакет запроса пакетов данных (16байт)

Тип пакета: `Downlink`

Размер пакета: `128 бит (16байт)`

Категория пакета: `SMP-M. Downlink. Пакеты показаний прибора`

Внутреннее имя: `smpm_dl_device_energy_16b_get_data`

Описание

Пакет запроса данных

pack_id - это номер пакета, строго число, без учета флагов DFF как в ID пакетов

Все пакеты с данными имеющие метку времени (относительную или точную) могут быть запрошены по запросу этим пакетом

Прибор игнорирует ответ на этот пакет в случае: - В случае если прибор не поддерживает запрашиваемый тип пакета - Нет данных на запрашиваемую дату (или дата не валидна)

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	385 (*bin* **001 10000001**)
year	год	7 бит	u7	2000	2127	-	-
month	месяц	4 бит	SmpmDlDeviceEnergy16bGetDataDataRequestMonth	SmpmDlDeviceEnergy16bGetDataDataRequestMonth.JAN	SmpmDlDeviceEnergy16bGetDataDataRequestMonth.DEC	-	-
day	день.	5 бит	u5	0	31	-	-
request_data_pack_ids	-	84 бит	tuple[SmpmDlDeviceEnergy16bGetDataDataRequestId;6]	-	-	-	-
↳ **pack_id**	-	14 бит	SmpmDlDeviceEnergy16bGetDataDataRequestId	SmpmDlDeviceEnergy16bGetDataDataRequestId.UNDEFINED	SmpmDlDeviceEnergy16bGetDataDataRequestId.NETWORK_PARAMS_PHASE1	-	-
RESERVED	для downlink-пакетов резервное место заполняется рандомными битами	17 бит	-	-	-	-	

Референсные значения типов данных

year

год

> Целое Беззнаковое

- Минимальное значение: `2000`
- Максимальное значение: `2127`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`

- тип данных: `u7`

month

месяц

- Минимальное значение: `SmpmDlDeviceEnergy16bGetDataDataRequestMonth.JAN`
- Максимальное значение: `SmpmDlDeviceEnergy16bGetDataDataRequestMonth.DEC`
- Размер: `4 бит`
- тип данных: `SmpmDlDeviceEnergy16bGetDataDataRequestMonth`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	-- Зарезервировано --
1	JAN
2	FEB
3	MAR
4	APR
5	MAY
6	JUN
7	JUL
8	AUG
9	SEP
10	OKT
11	NOV
12	DEC
13	-- Зарезервировано --
14	-- Зарезервировано --
15	-- Зарезервировано --

day

день.
(если стоит 0 значит текущий день и текущие показания, год и месяц в таком случае игнорируются)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `31`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `5 бит`
- тип данных: `u5`

request_data_pack_ids

- Количество элементов: `6`
- Размер: `84 бит`
- тип данных: `tuple[SmpmDlDeviceEnergy16bGetDataDataRequestId;6]`

pack_id

- Минимальное значение: `SmpmDlDeviceEnergy16bGetDataDataRequestId.UNDEFINED`
- Максимальное значение: `SmpmDlDeviceEnergy16bGetDataDataRequestId.NETWORK_PARAMS_PHASE1`
- Размер: `14 бит`
- тип данных: `SmpmDlDeviceEnergy16bGetDataDataRequestId`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	UNDEFINED
315	UL_DATA_16B__ENERGY
444	NETWORK_PARAMS_PHASE1
400	DAILY_ENERGY_ACTIVE_CONSUMED
401	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1
402	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2
403	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3
404	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4
405	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM
406	DAILY_ENERGY_REACTIVE_CONSUMED
407	DAILY_ENERGY_ACTIVE_GENERATED
408	DAILY_ENERGY_REACTIVE_GENERATED
409	MONTHLY_ENERGY_ACTIVE_CONSUMED
410	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1
411	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2
412	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3
413	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4
414	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM
415	MONTHLY_ENERGY_REACTIVE_CONSUMED
416	MONTHLY_ENERGY_ACTIVE_GENERATED
417	MONTHLY_ENERGY_REACTIVE_GENERATED

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyDlPackId.GET_DATA_LONG(129)
year: 2032
month: SmpmDlDeviceEnergy16bGetDataDataRequestMonth.JAN(1)
day: 15
request_data_pack_ids:

  - SmpmDlDeviceEnergy16bGetDataDataRequestId.MONTHLY_ENERGY_ACTIVE_CONSUMED(409)
  - SmpmDlDeviceEnergy16bGetDataDataRequestId.MONTHLY_ENERGY_REACTIVE_CONSUMED(415)
  - SmpmDlDeviceEnergy16bGetDataDataRequestId.DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3(403)
  - SmpmDlDeviceEnergy16bGetDataDataRequestId.DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM(405)
```

```
  - SmpmDlDeviceEnergy16bGetDataDataRequestId.NETWORK_PARAMS_PHASE1(444)
  - SmpmDlDeviceEnergy16bGetDataDataRequestId.UNDEFINED(0)
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	1	0000001
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	1	01
packet_type_id.1.DFF	bool	1	0	0
year	u7	7	32	0100000
month	SmpmDlDeviceEnergy16bGetDataDataRequestMonth	4	1	0001
day	u5	5	15	01111
request_data_pack_ids.0.pack_id	SmpmDlDeviceEnergy16bGetDataDataRequestId	14	409	00000110011001
request_data_pack_ids.1.pack_id	SmpmDlDeviceEnergy16bGetDataDataRequestId	14	415	00000110011111
request_data_pack_ids.2.pack_id	SmpmDlDeviceEnergy16bGetDataDataRequestId	14	403	00000110010011
request_data_pack_ids.3.pack_id	SmpmDlDeviceEnergy16bGetDataDataRequestId	14	405	00000110010101
request_data_pack_ids.4.pack_id	SmpmDlDeviceEnergy16bGetDataDataRequestId	14	444	00000110111100
request_data_pack_ids.5.pack_id	SmpmDlDeviceEnergy16bGetDataDataRequestId	14	0	00000000000000
RESERVED	u17	17	0	00000000000000000

Result BIN (MSB - LSB):

```
00000000 00000000 00000000 00000000 00001101 11100000 00110010 10100000 11001001 10000011 00111110 00001100 11001011 11000101 00000001 10000001
```

Result HEX (little-endian):

```
81 01 c5 cb 0c 3e 83 c9 a0 32 e0 0d 00 00 00 00
```

Парсинг/Сериализация значений

Python C

```python
from enum import IntEnum, unique
import math
import random
import typing


# PACKET (128 bits)   smpm_dl_device_energy_16b_get_data
#
# RESULT int:        4294343595594875952597107073
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00001101 11100000 00110010 10100000 11001001 10000011 00111110 00001100 11001011 11000101 00000001 10000001   LSB
# RESULT hex:  LE    81 01 c5 cb 0c 3e 83 c9 a0 32 e0 0d 00 00 00 00
#
# name                          type                                      size  value(int)  data(bits)
# ----------------------------  ----------------------------------------  ----  ----------  ----------------------------------------------------------------------------------------------------
# packet_type_id.0.VAL          u7                                           7           1                                                                                                0000001
# packet_type_id.0.DFF          bool                                         1           1                                                                                               1
# packet_type_id.1.VAL          u2                                           2           1                                                                                              01
# packet_type_id.1.DFF          bool                                         1           0                                                                                             0
```

```python
# year                             u7                                                7          32                                                                                                               0100000
# month                            SmpmDlDeviceEnergy16bGetDataDataRequestMonth      4           1                                                                                                           0001
# day                              u5                                                5          15                                                                                                      01111
# request_data_pack_ids.0.pack_id  SmpmDlDeviceEnergy16bGetDataDataRequestId        14         409                                                                                        00000110011001
# request_data_pack_ids.1.pack_id  SmpmDlDeviceEnergy16bGetDataDataRequestId        14         415                                                                          00000110011111
# request_data_pack_ids.2.pack_id  SmpmDlDeviceEnergy16bGetDataDataRequestId        14         403                                                            00000110010011
# request_data_pack_ids.3.pack_id  SmpmDlDeviceEnergy16bGetDataDataRequestId        14         405                                              00000110010101
# request_data_pack_ids.4.pack_id  SmpmDlDeviceEnergy16bGetDataDataRequestId        14         444                                00000110111100
# request_data_pack_ids.5.pack_id  SmpmDlDeviceEnergy16bGetDataDataRequestId        14           0                  00000000000000
# RESERVED                         u17                                              17           0 00000000000000000
# Reserved space is filled with random bits in DOWNLINK packets !
@unique
class SmpmDlDeviceEnergy16bGetDataDataRequestId(IntEnum):
    UNDEFINED = 0
    UL_DATA_16B__ENERGY = 315
    NETWORK_PARAMS_PHASE1 = 444
    DAILY_ENERGY_ACTIVE_CONSUMED = 400
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 401
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 402
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 403
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 404
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 405
    DAILY_ENERGY_REACTIVE_CONSUMED = 406
    DAILY_ENERGY_ACTIVE_GENERATED = 407
    DAILY_ENERGY_REACTIVE_GENERATED = 408
    MONTHLY_ENERGY_ACTIVE_CONSUMED = 409
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 410
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 411
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 412
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 413
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 414
    MONTHLY_ENERGY_REACTIVE_CONSUMED = 415
    MONTHLY_ENERGY_ACTIVE_GENERATED = 416
    MONTHLY_ENERGY_REACTIVE_GENERATED = 417



@unique
class SmpmDlDeviceEnergy16bGetDataDataRequestMonth(IntEnum):
    JAN = 1
    FEB = 2
    MAR = 3
    APR = 4
    MAY = 5
    JUN = 6
    JUL = 7
    AUG = 8
    SEP = 9
    OKT = 10
    NOV = 11
    DEC = 12


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
```

```
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmDlDeviceEnergy16BGetDataData(typing.NamedTuple):
    year: int
    month: SmpmDlDeviceEnergy16bGetDataDataRequestMonth
    day: int
    request_data_pack_ids: typing.Tuple[SmpmDlDeviceEnergy16bGetDataDataRequestId, SmpmDlDeviceEnergy16bGetDataDataRequestId, SmpmDlDeviceEnergy16bGetDataDataRequestId, SmpmDlDeviceEnergy16bGetDataDataRequestId, SmpmDlDeviceEnergy16bGetDataDataRequestId, SmpmDlDeviceEnergy16bGetDataDataRequestId]

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((1) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.year, int)
        assert 2000 <= data.year <= 2127
        result |= (((data.year + -2000)) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.month, SmpmDlDeviceEnergy16bGetDataDataRequestMonth)
        result |= ((data.month.value) & (2 ** (4) - 1)) << size
        size += 4
        assert isinstance(data.day, int)
        assert 0 <= data.day <= 31
        result |= ((data.day) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.request_data_pack_ids, tuple) and len(data.request_data_pack_ids) == 6
        assert isinstance(data.request_data_pack_ids[0], SmpmDlDeviceEnergy16bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[0].value) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.request_data_pack_ids[1], SmpmDlDeviceEnergy16bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[1].value) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.request_data_pack_ids[2], SmpmDlDeviceEnergy16bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[2].value) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.request_data_pack_ids[3], SmpmDlDeviceEnergy16bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[3].value) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.request_data_pack_ids[4], SmpmDlDeviceEnergy16bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[4].value) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.request_data_pack_ids[5], SmpmDlDeviceEnergy16bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[5].value) & (2 ** (14) - 1)) << size
        size += 14
        result |= ((random.getrandbits(17)) & (2 ** 17 - 1)) << size
        size += 17
        return result.to_bytes(16, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmDlDeviceEnergy16BGetDataData':
        result__el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 1 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(2):
```

```
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp1["year"] = buf.shift(7) + 2000
        result__el_tmp1["month"] = SmpmDlDeviceEnergy16bGetDataDataRequestMonth(buf.shift(4))
        result__el_tmp1["day"] = buf.shift(5) + 0
        request_data_pack_ids_tmp2: typing.List[SmpmDlDeviceEnergy16bGetDataDataRequestId] = []
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy16bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy16bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy16bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy16bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy16bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy16bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        result__el_tmp1["request_data_pack_ids"] = tuple(request_data_pack_ids_tmp2)
        result = SmpmDlDeviceEnergy16BGetDataData(**result__el_tmp1)
        buf.shift(17)
        return result
```

Последнее обновление: 2024-10-24

[Счетчики Электроэнергии] Пакет запроса пакетов данных укороченный (8байт)

Тип пакета: `Downlink`

Размер пакета: `64 бит (8байт)`

Категория пакета: `SMP-M. Downlink. Пакеты показаний прибора`

Внутреннее имя: `smpm_dl_device_energy_8b_get_data`

Описание

смотри пакет smpm_dl_device_energy_16b_get_data. такой же пакет но на больше пакетов можно запросить за раз

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	384 (*bin* **001 10000000**)
year	год	7 бит	u7	2000	2127	-	-
month	месяц	4 бит	SmpmDlDeviceEnergy8bGetDataDataRequestMonth	SmpmDlDeviceEnergy8bGetDataDataRequestMonth.JAN	SmpmDlDeviceEnergy8bGetDataDataRequestMonth.DEC	-	-
day	день.	5 бит	u5	0	31	-	-
request_data_pack_ids	-	28 бит	tuple[SmpmDlDeviceEnergy8bGetDataDataRequestId;2]	-	-	-	-
↳ **pack_id**	-	14 бит	SmpmDlDeviceEnergy8bGetDataDataRequestId	SmpmDlDeviceEnergy8bGetDataDataRequestId.UNDEFINED	SmpmDlDeviceEnergy8bGetDataDataRequestId.NETWORK_PARAMS_PHASE1	-	-
RESERVED	для downlink-пакетов резервное место заполняется рандомными битами	9 бит	-	-	-	-	

Референсные значения типов данных

year

год

> Целое Беззнаковое

- Минимальное значение: `2000`
- Максимальное значение: `2127`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `7 бит`
- тип данных: `u7`

month

месяц

- Минимальное значение: `SmpmDlDeviceEnergy8bGetDataDataRequestMonth.JAN`
- Максимальное значение: `SmpmDlDeviceEnergy8bGetDataDataRequestMonth.DEC`
- Размер: `4 бит`
- тип данных: `SmpmDlDeviceEnergy8bGetDataDataRequestMonth`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	-- Зарезервировано --
1	JAN
2	FEB
3	MAR
4	APR
5	MAY
6	JUN
7	JUL
8	AUG
9	SEP
10	OKT
11	NOV
12	DEC
13	-- Зарезервировано --
14	-- Зарезервировано --
15	-- Зарезервировано --

day

день.
(если стоит 0 значит текущий день и текущие показания, год и месяц в таком случае игнорируются)

> Целое Беззнаковое

- Минимальное значение: `0`
- Максимальное значение: `31`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `5 бит`
- тип данных: `u5`

request_data_pack_ids

- Количество элементов: `2`
- Размер: `28 бит`
- тип данных: `tuple[SmpmDlDeviceEnergy8bGetDataDataRequestId;2]`

pack_id

- Минимальное значение: `SmpmDlDeviceEnergy8bGetDataDataRequestId.UNDEFINED`
- Максимальное значение: `SmpmDlDeviceEnergy8bGetDataDataRequestId.NETWORK_PARAMS_PHASE1`
- Размер: `14 бит`
- тип данных: `SmpmDlDeviceEnergy8bGetDataDataRequestId`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	UNDEFINED
315	UL_DATA_16B__ENERGY
444	NETWORK_PARAMS_PHASE1
400	DAILY_ENERGY_ACTIVE_CONSUMED
401	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1
402	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2
403	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3
404	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4
405	DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM
406	DAILY_ENERGY_REACTIVE_CONSUMED
407	DAILY_ENERGY_ACTIVE_GENERATED
408	DAILY_ENERGY_REACTIVE_GENERATED
409	MONTHLY_ENERGY_ACTIVE_CONSUMED
410	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1
411	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2
412	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3
413	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4
414	MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM
415	MONTHLY_ENERGY_REACTIVE_CONSUMED
416	MONTHLY_ENERGY_ACTIVE_GENERATED
417	MONTHLY_ENERGY_REACTIVE_GENERATED

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyDlPackId.GET_DATA_SHORT(128)
year: 2032
month: SmpmDlDeviceEnergy8bGetDataDataRequestMonth.JAN(1)
day: 15
request_data_pack_ids:

  - SmpmDlDeviceEnergy8bGetDataDataRequestId.DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2(402)
  - SmpmDlDeviceEnergy8bGetDataDataRequestId.NETWORK_PARAMS_PHASE1(444)
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	0	0000000

name	type	size	value(int)	data(bits)
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	1	01
packet_type_id.1.DFF	bool	1	0	0
year	u7	7	32	0100000
month	SmpmDlDeviceEnergy8bGetDataDataRequestMonth	4	1	0001
day	u5	5	15	01111
request_data_pack_ids.0.pack_id	SmpmDlDeviceEnergy8bGetDataDataRequestId	14	402	00000110010010
request_data_pack_ids.1.pack_id	SmpmDlDeviceEnergy8bGetDataDataRequestId	14	444	00000110111100
RESERVED	u9	9	0	000000000

Result BIN (MSB - LSB):

`00000000 00000011 01111000 00001100 10010011 11000101 00000001 10000000`

Result HEX (little-endian):

```
80 01 c5 93 0c 78 03 00
```

Парсинг/Сериализация значений

Python   C

```python
from enum import IntEnum, unique
import math
import random
import typing


# PACKET (64 bits)   smpm_dl_device_energy_8b_get_data
#
# RESULT int:        976420344234368
# RESULT bin:  MSB   00000000 00000011 01111000 00001100 10010011 11000101 00000001 10000000   LSB
# RESULT hex:  LE    80 01 c5 93 0c 78 03 00
#
# name                             type                                         size  value(int)                                                        data(bits)
# -------------------------------  -------------------------------------------  ----  ----------  ----------------------------------------------------------------
# packet_type_id.0.VAL             u7                                              7           0                                                           0000000
# packet_type_id.0.DFF             bool                                            1           1                                                          1
# packet_type_id.1.VAL             u2                                              2           1                                                        01
# packet_type_id.1.DFF             bool                                            1           0                                                       0
# year                             u7                                              7          32                                                0100000
# month                            SmpmDlDeviceEnergy8bGetDataDataRequestMonth     4           1                                            0001
# day                              u5                                              5          15                                       01111
# request_data_pack_ids.0.pack_id  SmpmDlDeviceEnergy8bGetDataDataRequestId       14         402                         00000110010010
# request_data_pack_ids.1.pack_id  SmpmDlDeviceEnergy8bGetDataDataRequestId       14         444           00000110111100
# RESERVED                         u9                                              9           0  000000000
# Reserved space is filled with random bits in DOWNLINK packets !
@unique
class SmpmDlDeviceEnergy8bGetDataDataRequestId(IntEnum):
    UNDEFINED = 0
    UL_DATA_16B__ENERGY = 315
    NETWORK_PARAMS_PHASE1 = 444
    DAILY_ENERGY_ACTIVE_CONSUMED = 400
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 401
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 402
```

```
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 403
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 404
    DAILY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 405
    DAILY_ENERGY_REACTIVE_CONSUMED = 406
    DAILY_ENERGY_ACTIVE_GENERATED = 407
    DAILY_ENERGY_REACTIVE_GENERATED = 408
    MONTHLY_ENERGY_ACTIVE_CONSUMED = 409
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_1 = 410
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_2 = 411
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_3 = 412
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_4 = 413
    MONTHLY_ENERGY_ACTIVE_CONSUMED_TARIFF_SUM = 414
    MONTHLY_ENERGY_REACTIVE_CONSUMED = 415
    MONTHLY_ENERGY_ACTIVE_GENERATED = 416
    MONTHLY_ENERGY_REACTIVE_GENERATED = 417



@unique
class SmpmDlDeviceEnergy8bGetDataDataRequestMonth(IntEnum):
    JAN = 1
    FEB = 2
    MAR = 3
    APR = 4
    MAY = 5
    JUN = 6
    JUL = 7
    AUG = 8
    SEP = 9
    OKT = 10
    NOV = 11
    DEC = 12


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmDlDeviceEnergy8BGetDataData(typing.NamedTuple):
    year: int
    month: SmpmDlDeviceEnergy8bGetDataDataRequestMonth
    day: int
    request_data_pack_ids: typing.Tuple[SmpmDlDeviceEnergy8bGetDataDataRequestId, SmpmDlDeviceEnergy8bGetDataDataRequestId]

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
```

```
        result |= ((0) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.year, int)
        assert 2000 <= data.year <= 2127
        result |= (((data.year + -2000)) & (2 ** (7) - 1)) << size
        size += 7
        assert isinstance(data.month, SmpmDlDeviceEnergy8bGetDataDataRequestMonth)
        result |= ((data.month.value) & (2 ** (4) - 1)) << size
        size += 4
        assert isinstance(data.day, int)
        assert 0 <= data.day <= 31
        result |= ((data.day) & (2 ** (5) - 1)) << size
        size += 5
        assert isinstance(data.request_data_pack_ids, tuple) and len(data.request_data_pack_ids) == 2
        assert isinstance(data.request_data_pack_ids[0], SmpmDlDeviceEnergy8bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[0].value) & (2 ** (14) - 1)) << size
        size += 14
        assert isinstance(data.request_data_pack_ids[1], SmpmDlDeviceEnergy8bGetDataDataRequestId)
        result |= ((data.request_data_pack_ids[1].value) & (2 ** (14) - 1)) << size
        size += 14
        result |= ((random.getrandbits(9)) & (2 ** 9 - 1)) << size
        size += 9
        return result.to_bytes(8, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmDlDeviceEnergy8BGetDataData':
        result__el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 0 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp1["year"] = buf.shift(7) + 2000
        result__el_tmp1["month"] = SmpmDlDeviceEnergy8bGetDataDataRequestMonth(buf.shift(4))
        result__el_tmp1["day"] = buf.shift(5) + 0
        request_data_pack_ids_tmp2: typing.List[SmpmDlDeviceEnergy8bGetDataDataRequestId] = []
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy8bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        request_data_pack_ids__item_tmp3 = SmpmDlDeviceEnergy8bGetDataDataRequestId(buf.shift(14))
        request_data_pack_ids_tmp2.append(request_data_pack_ids__item_tmp3)
        result__el_tmp1["request_data_pack_ids"] = tuple(request_data_pack_ids_tmp2)
        result = SmpmDlDeviceEnergy8BGetDataData(**result__el_tmp1)
        buf.shift(9)
        return result
```

Последнее обновление: 2024-10-24

[Счетчики Электроэнергии] Установка Времени (8байт)

Тип пакета: `Downlink`

Размер пакета: `64 бит (8байт)`

Категория пакета: `SMP-M. Downlink. Пакеты показаний прибора`

Внутреннее имя: `smpm_dl_device_energy_8b_set_clock`

Описание

.

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	8 бит	const(dff(8,3,3,3...))	-	-	-	2 (*hex* **02**)
time	количество целых секунд с 2020-01-01 00:00:00	32 бит	timedelta	0 секунд	49710 дней 6 часов 28 минут 15 секунд	-	-
time_zone_offset_s	-	17 бит	timedelta	0 секунд	1 день 12 часа 24 минуты 31 секунда	-	-
time_zone_offset_is_negative	-	1 бит	bool	0	1	-	-
RESERVED	для downlink-пакетов резервное место заполняется рандомными битами	6 бит	-	-	-	-	

Референсные значения типов данных

time

количество целых секунд с 2020-01-01 00:00:00

- Гранулярность данных: `1 секунда`
- Минимальное значение: `0 секунд`
- Максимальное значение: `49710 дней 6 часов 28 минут 15 секунд`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`
- Размер: `32 бит`
- тип данных: `timedelta`

time_zone_offset_s

- Гранулярность данных: `1 секунда`
- Минимальное значение: `0 секунд`
- Максимальное значение: `1 день 12 часа 24 минуты 31 секунда`
- Лимитирование значения: `нет`
- Переполнение значения: `Не допустимо`

- Размер: `17 бит`
- тип данных: `timedelta`

time_zone_offset_is_negative

- Размер: `1 бит`
- тип данных: `bool`

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyDlPackId.SET_CLOCK(2)
time: 2147483647.0s
time_zone_offset_s: 65535.0s
time_zone_offset_is_negative: false
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	2	0000010
packet_type_id.0.DFF	bool	1	0	0
time	timedelta	32	2147483647	01111111111111111111111111111111
time_zone_offset_s	timedelta	17	65535	01111111111111111
time_zone_offset_is_negative	bool	1	0	0
RESERVED	u6	6	0	000000

Result BIN (MSB - LSB):

```
00000000 11111111 11111111 01111111 11111111 11111111 11111111 00000010
```

Result HEX (little-endian):

```
02 ff ff ff 7f ff ff 00
```

Парсинг/Сериализация значений

Python C

```python
from datetime import timedelta
import math
import random
import typing


# PACKET (64 bits)   smpm_dl_device_energy_8b_set_clock
#
# RESULT int:        72057044282113794
# RESULT bin:  MSB   00000000 11111111 11111111 01111111 11111111 11111111 11111111 00000010   LSB
# RESULT hex:  LE    02 ff ff ff 7f ff ff 00
#
```

```
# name                          type       size  value(int)                                                  data(bits)
# ----------------------------  ---------  ----  ----------  ----------------------------------------------------------------
# packet_type_id.0.VAL          u7            7           2                                                    0000010
# packet_type_id.0.DFF          bool          1           0                                                   0
# time                          timedelta    32  2147483647                       01111111111111111111111111111111
# time_zone_offset_s            timedelta    17       65535         01111111111111111
# time_zone_offset_is_negative  bool          1           0        0
# RESERVED                      u6            6           0  000000
# Reserved space is filled with random bits in DOWNLINK packets !


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size


class SmpmDlDeviceEnergy8BSetClockData(typing.NamedTuple):
    time: timedelta
    time_zone_offset_s: timedelta
    time_zone_offset_is_negative: bool

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((2) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        isinstance(data.time, (int, timedelta))
        time_tmp1 = int(data.time.total_seconds() // 1 if isinstance(data.time, timedelta) else data.time // 1)
        assert 0 <= time_tmp1 <= 4294967295
        result |= ((time_tmp1) & (2 ** (32) - 1)) << size
        size += 32
        isinstance(data.time_zone_offset_s, (int, timedelta))
        time_zone_offset_s_tmp2 = int(data.time_zone_offset_s.total_seconds() // 1 if isinstance(data.time_zone_offset_s, timedelta) else data.time_zone_offset_s // 1)
        assert 0 <= time_zone_offset_s_tmp2 <= 131071
        result |= ((time_zone_offset_s_tmp2) & (2 ** (17) - 1)) << size
        size += 17
        assert isinstance(data.time_zone_offset_is_negative, bool)
        result |= ((int(data.time_zone_offset_is_negative)) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((random.getrandbits(6)) & (2 ** 6 - 1)) << size
        size += 6
        return result.to_bytes(8, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmDlDeviceEnergy8BSetClockData':
        result__el_tmp3: typing.Dict[str, typing.Any] = dict()
```

```
        if 2 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp3["time"] = timedelta(seconds=buf.shift(32) * 1)
        result__el_tmp3["time_zone_offset_s"] = timedelta(seconds=buf.shift(17) * 1)
        result__el_tmp3["time_zone_offset_is_negative"] = bool(buf.shift(1))
        result = SmpmDlDeviceEnergy8BSetClockData(**result__el_tmp3)
        buf.shift(6)
        return result
```

Последнее обновление: 2024-08-27

[Счетчики Электроэнергии] Установка состояния реле

Тип пакета: `Downlink`

Размер пакета: `64 бит (8байт)`

Категория пакета: `SMP-M. Downlink. Пакеты показаний прибора`

Внутреннее имя: `smpm_dl_device_energy_8b_set_relay`

Описание

Установка реле (относительно пропускаемого через него тока)

В ответ на этот пакет прибор гарантирует отправку любого пакета, где есть указание состояние реле (тока) после установки этим пакетом.

Состав пакета

Поле	Описание	Размер	Тип	Мин	Макс	Переполнение	Const значение (bin/hex)
packet_type_id	идентификатор пакета	11 бит	const(dff(8,3,3,3...))	-	-	-	426 (*bin* **001 10101010**)
state	Состояние реле (протекаемого через него тока).	1 бит	bool	0	1	-	-
relay_id	-	8 бит	SmpmDlDeviceEnergy8bSetRelayId	SmpmDlDeviceEnergy8bSetRelayId.ALL	SmpmDlDeviceEnergy8bSetRelayId.RELAY_ID_7	-	-
RESERVED	для downlink-пакетов резервное место заполняется рандомными битами	44 бит	-	-	-	-	

Референсные значения типов данных

state

Состояние реле (протекаемого через него тока).
1 - ток идет.
0 - не идет.

- Размер: `1 бит`
- тип данных: `bool`

relay_id

- Минимальное значение: `SmpmDlDeviceEnergy8bSetRelayId.ALL`
- Максимальное значение: `SmpmDlDeviceEnergy8bSetRelayId.RELAY_ID_7`
- Размер: `8 бит`

- тип данных: `SmpmDlDeviceEnergy8bSetRelayId`

Может принимать одно значение из приведенного ниже списка:

Значение	Имя
0	ALL
1	RELAY_ID_1
2	RELAY_ID_2
3	RELAY_ID_3
4	RELAY_ID_4
5	RELAY_ID_5
6	RELAY_ID_6
7	RELAY_ID_7

Пример пакета

```
packet_type_id: UpLinkDeviceEnergyDlPackId.SET_RELAY(170)
state: false
relay_id: SmpmDlDeviceEnergy8bSetRelayId.ALL(0)
```

Результат

name	type	size	value(int)	data(bits)
packet_type_id.0.VAL	u7	7	42	0101010
packet_type_id.0.DFF	bool	1	1	1
packet_type_id.1.VAL	u2	2	1	01
packet_type_id.1.DFF	bool	1	0	0
state	bool	1	0	0
relay_id	SmpmDlDeviceEnergy8bSetRelayId	8	0	00000000
RESERVED	u44	44	0	00000000000000000000000000000000000000000000

Result BIN (MSB - LSB):

```
 00000000 00000000 00000000 00000000 00000000 00000000 00000000 10101010
```

Result HEX (little-endian):

```
aa 01 00 00 00 00 00 00
```

Парсинг/Сериализация значений

Python C

```python
from enum import IntEnum, unique
import math
import random
import typing


# PACKET (64 bits)   smpm_dl_device_energy_8b_set_relay
#
# RESULT int:        426
# RESULT bin:  MSB   00000000 00000000 00000000 00000000 00000000 00000000 00000001 10101010   LSB
# RESULT hex:  LE    aa 01 00 00 00 00 00 00
#
# name                  type                           size  value(int)                                                  data(bits)
# --------------------  -----------------------------  ----  ----------  ----------------------------------------------------------------
# packet_type_id.0.VAL  u7                               7          42                                                     0101010
# packet_type_id.0.DFF  bool                             1           1                                                    1
# packet_type_id.1.VAL  u2                               2           1                                                   01
# packet_type_id.1.DFF  bool                             1           0                                                  0
# state                 bool                             1           0                                                 0
# relay_id              SmpmDlDeviceEnergy8bSetRelayId     8           0                                         00000000
# RESERVED              u44                               44           0  00000000000000000000000000000000000000000000
# Reserved space is filled with random bits in DOWNLINK packets !
@unique
class SmpmDlDeviceEnergy8bSetRelayId(IntEnum):
    ALL = 0
    RELAY_ID_1 = 1
    RELAY_ID_2 = 2
    RELAY_ID_3 = 3
    RELAY_ID_4 = 4
    RELAY_ID_5 = 5
    RELAY_ID_6 = 6
    RELAY_ID_7 = 7


class BufRef:
    def __init__(self, data: typing.Union[bytes, bytearray, int], starts_at: int = 0, stop_on_buffer_end: bool = False) -> None:
        self.value: typing.Union[bytes, bytearray] = data if not isinstance(data, int) else data.to_bytes(math.ceil(max(data.bit_length(), 1) / 8), "little")
        self._stop_on_buffer_end, self.ends_at = stop_on_buffer_end, starts_at

    def shift(self, size: int) -> int:
        bytes_i, bit_i = self.ends_at // 8, self.ends_at % 8
        res_size, res, needable_len = 0, 0, bytes_i + math.ceil((bit_i + size) / 8)
        buf_len = len(self.value)
        for i in range(bytes_i, needable_len):
            if i >= buf_len:
                if self._stop_on_buffer_end: break
                raise ValueError(f'buffer has no enough elements. {buf_len} < {needable_len}')
            s = min(8 - bit_i, size - res_size)
            res |= ((self.value[i] >> bit_i) & (2 ** s - 1)) << res_size
            res_size += s
            bit_i = 0
        self.ends_at += res_size
        return res

    def offset(self, size: int) -> None:
        assert size >= 0, f'{size} was given'
        self.ends_at += size
```

```
class SmpmDlDeviceEnergy8BSetRelayData(typing.NamedTuple):
    state: bool
    relay_id: SmpmDlDeviceEnergy8bSetRelayId

    def serialize(self) -> bytes:
        data = self
        result = 0
        size = 0
        result |= ((42) & (2 ** (7) - 1)) << size
        size += 7
        result |= ((1) & (2 ** (1) - 1)) << size
        size += 1
        result |= ((1) & (2 ** (2) - 1)) << size
        size += 2
        result |= ((0) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.state, bool)
        result |= ((int(data.state)) & (2 ** (1) - 1)) << size
        size += 1
        assert isinstance(data.relay_id, SmpmDlDeviceEnergy8bSetRelayId)
        result |= ((data.relay_id.value) & (2 ** (8) - 1)) << size
        size += 8
        result |= ((random.getrandbits(44)) & (2 ** 44 - 1)) << size
        size += 44
        return result.to_bytes(8, "little")

    @classmethod
    def parse(cls, buf: BufRef) -> 'SmpmDlDeviceEnergy8BSetRelayData':
        result__el_tmp1: typing.Dict[str, typing.Any] = dict()
        if 42 != buf.shift(7):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 1 != buf.shift(2):
            raise ValueError("packet_type_id: buffer doesn't match value")
        if 0 != buf.shift(1):
            raise ValueError("packet_type_id: buffer doesn't match value")
        result__el_tmp1["state"] = bool(buf.shift(1))
        result__el_tmp1["relay_id"] = SmpmDlDeviceEnergy8bSetRelayId(buf.shift(8))
        result = SmpmDlDeviceEnergy8BSetRelayData(**result__el_tmp1)
        buf.shift(44)
        return result
```

Последнее обновление: 2024-08-27