Kamstrup OmniPower wm-bus metering

Release development

Team 3, E5PRO5 2020, Aarhus Uni., School of Engineering

Oct 16, 2020

TABLE OF CONTENTS

CHAPTER

ONE

OMNIPOWER IMPLEMENTATION

Parse Kamstrup OmniPower wm-bus telegrams

platform Python 3.5.10 on Linux, OS X

synopsis Implements parsing functionality for C1 telegrams and log handling for data series

author Janus Bo Andersen

date 14 October 2020

- This module implements parsing for the Kamstrup OmniPower meter, single-phase.
- This meter sends wm-bus C1 (compact one-way) telegrams.
- Telegrams on wm-bus are little-endian, i.e. LSB first.

In a regular measurement data telegram, the data fields are:

#	Byte#	Bytes	M-bus field	Description	Expected value (little-endian)
0	0	1	L	Telegram length	0x27 (39 bytes follow)
1	1	1	C	Control field (type and purpose of message)	0x44 (SND_NR)
2	2-3	2	M	Manufacturer ID (official ID code)	0x2D2C (KAM)
3	4-7	4	A	Address (meter serial number)	0x57686632 (big endian 32666857)
4	8	1	Ver.	Version number of the wm-bus firmware	0x30
5	9	1	Medium	Type / medium of meter	0x02 (Electricity)
6	10	1	CI	Control Information	0x8D (Extended Link Layer 2)
7	11	1	CC	Communication Control	0x20 (Slow response sync.)
8	12	1	ACC	Access field	Varies
9	13-16	4	AES_CTR	AES counter	Varies, used for decryption
10	17-39	23	Data	Contains AES-encrypted data frame	Encrypted data
11	40-41	2	CRC16	CRC16 check	

The first 9 fields of the telegram can be unpacked using the little-endian format *<BBHIBBBBB*, where

- *<* marks little-endian,
- *B* is an unsigned 1 byte (char),
- *H* is an unsigned 2 byte (short),
- *I* is an unsigned 4 byte (int)

Telegram examples

L	C	M	A	Ver	Med	CI	CC	ACC	AES CTR	Encrypted payload	CRC 16
27	44	2D 2C	5768 6632	30	02	8D	20	2E	2187 0320	D3A4F149 B1B8F578 3DF7434B 8A66A557 86499ABE 7BAB59	xxxx
27	44	2d 2c	5768 6632	30	02	8d	20	63	60dd 0320	c42b87f4 6fc048d4 2498b44b 5e34f083 e93e6af1 617631	3d9c
27	44	2d 2c	5768 6632	30	02	8d	20	8e	11de 0320	188851bd c4b72dd3 c2954a34 1be369e9 089b4eb3 858169	494e

The AES-CTR decryption prefix is built from some of the fields (m-bus mode 5). It's packed using the format *<HI-BBBIB*.

M	A	Ver	Med	CC	AES_CTR	Pad
2D2C	57686632	30	02	20	21870320	00

A decrypted payload example

CRC16	CI-TPL	Data fmt. sign.	CRC16 data	Field 1	Field 2	Field 3	Field 4
1170	79	138C	4491	CE000000	00000000	03000000	00000000

Measurement fields start at byte 7, and can be extracted using *<IIII* little-endian format.

The fields in the OmniPower are

Field	Kamstrup name	Data fmt (DIF)	Value type (VIF/E)	VIF/E meaning	DIF VIF/E
Field 1	A+	32-bit uint	Energy, 10^1 Wh	Consumption from grid, accum.	04 04
Field 2	A-	32-bit uint	Energy, 10^1 Wh	Production to grid, accum.	04 84 3C
Field 3	P+	32-bit uint	Power, 10^0 W	Consumption from grid, instantan.	04 2B
Field 4	P-	32-bit uint	Power, 10^0 W	Production to grid, instantan.	04 AB 3C

1.1 The C1 Telegram class

class OmniPower.OmniPower.**C1Telegram**(*telegram: bytes*)

Implements capture of data fields for a C1 telegram from OmniPower

decrypt_using(*meter:* OmniPower.OmniPower.OmniPower) → bool

Decrypts a telegram using the key from the specified meter. Updates the decrypted field of self. Requires instantiated OmniPower meter with valid AES-key.

1.2 The OmniPower class

class OmniPower.OmniPower.**OmniPower**(*name: str = 'Kamstrup OmniPower one-phase', meter_id: str = '32666857', manufacturer_id: str = '2C2D', medium: str = '02', version: str = '30', aes_key: str = '9A25139E3244CC2E391A8EF6B915B697'*)

Implementation of our OmniPower single-phase meter Passed values are hex encoded as string, e.g. '2C2D' for value 0x2C2D.

add_measurement_to_log(*measurement: OmniPower.MeterMeasurement.MeterMeasurement*) → None

Pushes a new measurement to the tail end of the log

decrypt(*telegram:* OmniPower.OmniPower.C1Telegram) → bytes

Decrypt a telegram. Requires:

- the prefix from the telegram (telegram.prefix), and
- the encryption key from the meter.

Decrypts the data stored telegram.encrypted

dump_log_to_json() → str

Returns a JSON object of all measurement frames in log, with an incremented number for each observation

extract_measurement_frame(*telegram:* OmniPower.OmniPower.C1Telegram) → OmniPower.MeterMeasurement.MeterMeasurement

Requires that the telegram is already decrypted, otherwise returns empty measurement

is_this_my(*telegram:* OmniPower.OmniPower.C1Telegram) → bool

Check whether a given telegram is from this meter by comparing meter setting to telegram

process_telegram(*telegram:* OmniPower.OmniPower.C1Telegram) → bool

Does entire processing chain for a telegram, including adding to log

classmethod unpack_long_telegram(*data: bytes*) → Tuple[int, ...]

Long C1 telegrams contain DIF/VIF information and field data values

classmethod unpack_short_telegram(*data: bytes*) → Tuple[int, ...]

Short C1 telegrams only contain field data values, no information about DIF/VIF

CHAPTER

TWO

IMPLEMENTATION OF GENERIC MEASUREMENTS

Generic class for measurements and measurement frames.

platform Python 3.5.10 on Linux, OS X

synopsis This module implements classes for generic measurements taken from a meter.

authors Janus Bo Andersen, Jakob Aaboe Vestergaard

date 13 October 2020

2.1 The Measurement class

class OmniPower.OmniPower.**Measurement**(*value: float*, *unit: str*)

Single physical measurement. A single measurement of a physical quantity pair, consisting of a value and a unit.

2.2 The MeterMeasurement class

class OmniPower.OmniPower.**MeterMeasurement**(*meter_id: str*, *timestamp: datetime.datetime*)

A single measurement collection based on one frame from the meter. Will contain multiple measurements of physical quantities taken at the same time.

add_measurement(*name: str*, *measurement: OmniPower.MeterMeasurement.Measurement*) → None

Store a new measurement in the collection.

as_dict() → dict

Serializes and dumps the Measurement frame as a dict. Make an object similar to {

"Meter ID: ": "3232323", "Timestamp:": "2020-10-13T17:36:53", "Measurements": {

"A+": { "unit": "kWh", "value": 7

}, "A-": {

"unit": "kWh", "value": 8

}, "P+": {

"unit": "kW", "value": 9

}, "P-": {

"unit": "kW", "value": 10

```
            }
        }
    }
```

json_dump() → str
Returns a JSON formatted string of all data in frame.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX