Decode MBUS-GEM

0.1.0

Generated on Fri Dec 2 2022 11:50:27 for Decode MBUS-GEM by Doxygen 1.9.5

Fri Dec 2 2022 11:50:27

1 mbusgemdecoder package

Decode MBUS-GEM register data into human-readable JSON.

1.1 Introduction

The goal of `mbusgemdecoder` package is to convert a list of ten integer values into human-readable data object. `mbusgemdecoder` package automatically detects the type (`MBUS-GEM gateway`, `METER`, `METER VALUE`) of the register(s) and parses the data accordingly.

However, `mbusgemdecoder` package is only for data conversion. Use, for example, `pyModbusTCP` to obtain data to convert with `mbusgemdecoder`.

2 Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

6 Namespace Documentation

6.1 mbus_gem_decoder Namespace Reference

Namespaces

- namespace conversion
- namespace mbusdecode

6.2 mbus_gem_decoder.conversion Namespace Reference

Namespaces

- namespace mbusmbus
- namespace mbusmeter
- namespace mbusmeterentry
- namespace utils

6.3 mbus_gem_decoder.conversion.mbusmbus Namespace Reference

Classes

- class MBusMBus

  MBUS-GEM gateway class.

6.4 mbus_gem_decoder.conversion.mbusmeter Namespace Reference

Classes

- class MBusMeter

  MBUS-GEM METER class.

6.5 mbus_gem_decoder.conversion.mbusmeterentry Namespace Reference

Classes

- class MBusMeterEntry

  MBUS-GEM METER ENTRY class.

6.6 mbus_gem_decoder.conversion.utils Namespace Reference

Namespaces

- namespace helpers
- namespace mbustypes

6.7 mbus_gem_decoder.conversion.utils.helpers Namespace Reference

Functions

- str meter_serial (list[int] ten_regs)
  Convert METER's serial number.
- str meter_manufacturer (list[int] ten_regs)
  Convert a numeric manufacturer code to string/letters.
- int meter_version (list[int] ten_regs)
  Convert METER's version to integer value.
- int meter_medium (list[int] ten_regs)
  METER's medium.
- str meter_medium_str (int value=-1)
  Convert numeric medium type to string.
- int meter_flag1 (list[int] ten_regs)
  Get METER's flag1 as integer.
- str meter_flag1_str (list[int] ten_regs)
  Get METER's flag1 as string.
- int meter_flag2 (list[int] ten_regs)
  Get METER's flag2 as integer.
- str meter_flag2_str (list[int] ten_regs)
  Get METER's flag2 as string.
- int register_type (list[int] ten_regs)
  METER's type as integer.
- str register_type_str (list[int] ten_regs)
  METER's type as string.
- int get_sign_correction (list[int] ten_regs)
  Get sign correction value.
- int get_integer_value (list[int] ten_regs)
  Get integer value of the reading.
- int get_scale (list[int] ten_regs)
  Scaling factor.
- float get_float_value (list[int] ten_regs)
  Get floating point value.
- float int32_to_ieee (int val_int)
  Convert Python int32 to IEEE float.
- float two_words_to_ieee (int val_one, int val_two)
  Convert two words (16 bit) to IEEE float (32 bit).
- int two_words_to_long (int val_one, int val_two)
  Convert two words (16 bit) to long (32 bit).
- int get_unit (list[int] ten_regs)
  Unit of measure.
- dict[str, str] get_unit_type (int value)
  Convert numeric unit type to string.
- str mbus_serial (list[int] ten_regs)
  Get serial number of MBUS-GEM gateway.
- int mbus_protocol_version (list[int] ten_regs)
  Protocol version.
- str mbus_version (list[int] ten_regs)
  Protocol version.
- int get_unix_timestamp (list[int] ten_regs)
  UNIX timestamp.
- str get_timestamp (list[int] ten_regs)
  Timestamp.

6.7.1 Function Documentation

6.7.1.1 get_float_value() `float mbus_gem_decoder.conversion.utils.helpers.get_float_value ( list[int] ten_regs )`

Get floating point value.

Parameters

ten_regs	registers

Returns

float approximate floating point value

Definition at line 232 of file helpers.py.

```
00232     """
00233
00234     return two_words_to_ieee(ten_regs[4], ten_regs[5])
00235
00236
```

6.7.1.2 get_integer_value() `int mbus_gem_decoder.conversion.utils.helpers.get_integer_value ( list[int] ten_regs )`

Get integer value of the reading.

Parameters

ten_regs	List of four (4) 16-bit integer values

Returns

int 64-bit integer value

Definition at line 198 of file helpers.py.

```
00198     """
00199
00200     sign_correction = get_sign_correction(ten_regs)
00201     raw_value = (ten_regs[0] & 0x3FFF) « 16
00202     raw_value = (raw_value + ten_regs[1]) « 16
00203     raw_value = (raw_value + ten_regs[2]) « 16
00204     raw_value = raw_value + ten_regs[3]
00205     return abs(sign_correction * raw_value) * sign_correction
00206
00207
```

6.7.1.3 get_scale() `int mbus_gem_decoder.conversion.utils.helpers.get_scale ( list[int] ten_regs )`

Scaling factor.

Parameters

ten_regs	registers

Returns

int scaling factor

$$float \approx integer \times 10^{scale}$$

Definition at line 218 of file helpers.py.

```
00218     """
00219
00220     correction = -1 if (ten_regs[6] » 15) & 0x01 else 1
00221     return abs((correction * ten_regs[6]) & 0x7FFF) * correction
00222
00223
```

6.7.1.4 get_sign_correction() `int mbus_gem_decoder.conversion.utils.helpers.get_sign_correction (`

```
            list[int] ten_regs )
```

Get sign correction value.

Parameters

ten_regs	registers

Returns

int sign correction (1 or -1)

Definition at line 185 of file helpers.py.

```
00185     """
00186
00187     return -1 if (ten_regs[0] » 15) & 0x01 else 1
00188
00189
```

6.7.1.5 get_timestamp() `str mbus_gem_decoder.conversion.utils.helpers.get_timestamp (`

```
            list[int] ten_regs )
```

Timestamp.

Parameters

ten_regs	registers

Returns

str Timestamp

Definition at line 379 of file helpers.py.

```
00379     """
00380
00381     unix_time = get_unix_timestamp(ten_regs)
00382     return datetime.utcfromtimestamp(unix_time).strftime("%Y-%m-%d %H:%M:%S")
```

6.7.1.6 get_unit()

`int mbus_gem_decoder.conversion.utils.helpers.get_unit ( list[int] ten_regs )`

Unit of measure.

Parameters

ten_regs	registers

Returns

int Unit of measure

Definition at line 287 of file helpers.py.

```
00287     """
00288
00289     return ten_regs[7] & 0xFF
00290
00291
```

6.7.1.7 get_unit_type()

`dict[str, str] mbus_gem_decoder.conversion.utils.helpers.get_unit_type ( int value )`

Convert numeric unit type to string.

Parameters

value	Numeric unit type

Returns

dict[str, str]: Unit type as a dictionary of strings (name (short form), desc (description or full form of the unit))

Definition at line 301 of file helpers.py.

```
00301     """
00302
00303     if value in UNIT_TYPES_ARRAY:
00304         unit_match = list(
00305             filter(
00306                 lambda x: x["num"] == value, UNIT_TYPES))
00307         return list(
00308             map(
00309                 lambda x:
```

```
00310                 {
00311                     "name": x["name"],
00312                     "desc": x["desc"]
00313                 }, unit_match))[0]
00314     if value < 0 or value > 255:
00315         raise Exception("Unit value out of range")
00316     return {"name": "Res", "desc": "Reserved"}
00317
00318
```

6.7.1.8 get_unix_timestamp() int mbus_gem_decoder.conversion.utils.helpers.get_unix_timestamp (list[int] *ten_regs*)

UNIX timestamp.

Parameters

ten_regs	registers

Returns

int UNIX timestamp

Definition at line 366 of file helpers.py.

```
00366     """
00367
00368     return (ten_regs[4] « 16) + ten_regs[5]
00369
00370
```

6.7.1.9 int32_to_ieee() float mbus_gem_decoder.conversion.utils.helpers.int32_to_ieee (int *val_int*)

Convert Python int32 to IEEE float.

Parameters

val_int	Value to convert to float

Returns

float Converted value as an IEEE floating point

Definition at line 245 of file helpers.py.

```
00245     """
00246
00247     return struct.unpack("f", struct.pack("I", val_int))[0]
00248
00249
```

6.7.1.10 mbus_protocol_version() int mbus_gem_decoder.conversion.utils.helpers.mbus_protocol_↩ version (

```
            list[int] ten_regs )
```

Protocol version.

Parameters

ten_regs	registers

Returns

int Protocol version

Definition at line 340 of file helpers.py.

```
00340     """
00341
00342     return ten_regs[2]
00343
00344
```

6.7.1.11 mbus_serial() str mbus_gem_decoder.conversion.utils.helpers.mbus_serial (

```
            list[int] ten_regs )
```

Get serial number of MBUS-GEM gateway.

Parameters

ten_regs	registesr

Returns

str serial number

Definition at line 327 of file helpers.py.

```
00327     """
00328
00329     return f"{ten_regs[0]:04X}{ten_regs[1]:04X}"
00330
00331
```

6.7.1.12 mbus_version() str mbus_gem_decoder.conversion.utils.helpers.mbus_version (

```
            list[int] ten_regs )
```

Protocol version.

Parameters

ten_regs	registers

Returns

str Protocol version

Definition at line 353 of file helpers.py.

```
00353     """
00354
00355     return f"{ten_regs[3]//100}.{ten_regs[3]%100}"
00356
00357
```

6.7.1.13 meter_flag1() `int mbus_gem_decoder.conversion.utils.helpers.meter_flag1 ( list[int] ten_regs )`

Get METER's flag1 as integer.

Parameters

ten_regs	registers

Returns

int flag1

Definition at line 101 of file helpers.py.

```
00101     """
00102
00103     return ten_regs[8] & 0x01
00104
00105
```

6.7.1.14 meter_flag1_str() `str mbus_gem_decoder.conversion.utils.helpers.meter_flag1_str ( list[int] ten_regs )`

Get METER's flag1 as string.

Parameters

ten_regs	registers

Returns

str flag1

Definition at line 114 of file helpers.py.

```
00114     """
00115
00116     return MeterFlag(ten_regs[8] & 0x01).name
00117
00118
```

6.7.1.15 meter_flag2() int mbus_gem_decoder.conversion.utils.helpers.meter_flag2 (list[int] *ten_regs*)

Get METER's flag2 as integer.

Parameters

ten_regs	registers

Returns

int flag2

Definition at line 127 of file helpers.py.

```
00127     """
00128
00129     return ((ten_regs[8] » 1) & 0x01) + 2
00130
00131
```

6.7.1.16 meter_flag2_str() str mbus_gem_decoder.conversion.utils.helpers.meter_flag2_str (list[int] *ten_regs*)

Get METER's flag2 as string.

Parameters

ten_regs	registers

Returns

str flag2

Definition at line 140 of file helpers.py.

```
00140     """
00141
00142     return MeterFlag(((ten_regs[8] » 1) & 0x01) + 2).name
00143
00144
```

6.7.1.17 meter_manufacturer() str mbus_gem_decoder.conversion.utils.helpers.meter_manufacturer (list[int] *ten_regs*)

Convert a numeric manufacturer code to string/letters.

Parameters

ten_regs	registers

Returns

str Three letter code of manufacturer

Definition at line 38 of file helpers.py.

```
00038     """
00039
00040     reg = ten_regs[2]
00041     id_1 = chr(ord("A") + ((reg » 10) & 0x1F) - 1)
00042     id_2 = chr(ord("A") + ((reg » 5) & 0x1F) - 1)
00043     id_3 = chr(ord("A") + (reg & 0x1F) - 1)
00044     return f"{id_1}{id_2}{id_3}"
00045
00046
```

6.7.1.18 meter_medium() int mbus_gem_decoder.conversion.utils.helpers.meter_medium (list[int] *ten_regs*)

METER's medium.

Parameters

ten_regs	registers

Returns

int medium

Definition at line 68 of file helpers.py.

```
00068     """
00069
00070     return ten_regs[3] & 0xFF
00071
00072
```

6.7.1.19 meter_medium_str() str mbus_gem_decoder.conversion.utils.helpers.meter_medium_str (int *value = -1*)

Convert numeric medium type to string.

Parameters

value	Numeric medium type. Defaults to -1.

Returns

str Medium type as a string

Definition at line 81 of file helpers.py.

```
00081     """
00082
00083     if value in MEDIUM_TYPES_ARRAY:
00084         return list(
```

```
00085             filter(
00086                 lambda x: x["num"] == value,
00087                 MEDIUM_TYPES))[0]["name"]
00088     if value < 0 or value > 255:
00089         raise Exception("Medium index is out of range")
00090     return "Reserved"
00091
00092
```

6.7.1.20 meter_serial() `str mbus_gem_decoder.conversion.utils.helpers.meter_serial ( list[int] ten_regs )`

Convert METER's serial number.

Parameters

ten_regs	registers

Returns

str serial number

Definition at line 25 of file helpers.py.

```
00025     """
00026
00027     return f"{((ten_regs[0] << 16) + ten_regs[1]):08d}"
00028
00029
```

6.7.1.21 meter_version() `int mbus_gem_decoder.conversion.utils.helpers.meter_version ( list[int] ten_regs )`

Convert METER's version to integer value.

Parameters

ten_regs	registers

Returns

int version number

Definition at line 55 of file helpers.py.

```
00055     """
00056
00057     return ten_regs[3] >> 8
00058
00059
```

6.7.1.22 register_type() `int mbus_gem_decoder.conversion.utils.helpers.register_type ( list[int] ten_regs )`

METER's type as integer.

Parameters

ten_regs	registers

Returns

int type

Definition at line 153 of file helpers.py.

```
00153     """
00154
00155     register_type_value = abs(ten_regs[7] » 8)
00156     if register_type_value not in REGISTER_TYPES_ARRAY:
00157         raise Exception("Register type value out of range")
00158     return register_type_value
00159
00160
```

6.7.1.23 register_type_str() `str mbus_gem_decoder.conversion.utils.helpers.register_type_str (list[int] ten_regs )`

METER's type as string.

Parameters

ten_regs	registers

Returns

str type

Definition at line 169 of file helpers.py.

```
00169     """
00170
00171     register_type_value = abs(ten_regs[7] » 8)
00172     if register_type_value not in REGISTER_TYPES_ARRAY:
00173         raise Exception("Register type value out of range")
00174     return RegisterType(register_type_value).name
00175
00176
```

6.7.1.24 two_words_to_ieee() `float mbus_gem_decoder.conversion.utils.helpers.two_words_to_ieee (int val_one, int val_two )`

Convert two words (16 bit) to IEEE float (32 bit).

Parameters

val_one	The first value (16 bit)
val_two	The second value (16 bit)

Returns

float Combined IEEE float (32 bit)

Definition at line 259 of file helpers.py.

```
00259     """
00260
00261     val_int32 = two_words_to_long(val_one, val_two)
00262     return int32_to_ieee(val_int32)
00263
00264
```

6.7.1.25 two_words_to_long()

```
int mbus_gem_decoder.conversion.utils.helpers.two_words_to_long (
        int val_one,
        int val_two )
```

Convert two words (16 bit) to long (32 bit).

Parameters

val_one	The first value (16 bit)
val_two	The second value (16 bit)

Returns

int Combined long (32 bit)

Definition at line 274 of file helpers.py.

```
00274     """
00275
00276     return (val_one « 16) + val_two
00277
00278
```

6.8 mbus_gem_decoder.conversion.utils.mbustypes Namespace Reference

Classes

- class MeterFlag

  Enumerator for meter status messages.
- class ReadabilityTypes
- class RegisterType

  Enumerator for register types.

Variables

- list MEDIUM_TYPES
- list UNIT_TYPES
- REGISTER_TYPES_ARRAY = list(map(lambda item: item.value, RegisterType))
- UNIT_TYPES_ARRAY = list(map(lambda item: item["num"], UNIT_TYPES))
- MEDIUM_TYPES_ARRAY = list(map(lambda item: item["num"], MEDIUM_TYPES))
- METER_TYPES_ARRAY = list(map(lambda item: item.value, MeterFlag))
- READABILITY_TYPES_ARRAY = list(map(lambda item: item.value, ReadabilityTypes))

6.8.1 Variable Documentation

6.8.1.1 MEDIUM_TYPES

`list mbus_gem_decoder.conversion.utils.mbustypes.MEDIUM_TYPES`

Definition at line 58 of file mbustypes.py.

6.8.1.2 MEDIUM_TYPES_ARRAY

`mbus_gem_decoder.conversion.utils.mbustypes.MEDIUM_TYPES_ARRAY = list(map(lambda item: item["num"], MEDIUM_TYPES))`

Definition at line 149 of file mbustypes.py.

6.8.1.3 METER_TYPES_ARRAY

`mbus_gem_decoder.conversion.utils.mbustypes.METER_TYPES_ARRAY = list(map(lambda item: item.value, MeterFlag))`

Definition at line 150 of file mbustypes.py.

6.8.1.4 READABILITY_TYPES_ARRAY

`mbus_gem_decoder.conversion.utils.mbustypes.READABILITY_↩ TYPES_ARRAY = list(map(lambda item: item.value, ReadabilityTypes))`

Definition at line 151 of file mbustypes.py.

6.8.1.5 REGISTER_TYPES_ARRAY

`mbus_gem_decoder.conversion.utils.mbustypes.REGISTER_TYPES_↩ ARRAY = list(map(lambda item: item.value, RegisterType))`

Definition at line 147 of file mbustypes.py.

6.8.1.6 UNIT_TYPES

`list mbus_gem_decoder.conversion.utils.mbustypes.UNIT_TYPES`

Definition at line 97 of file mbustypes.py.

6.8.1.7 UNIT_TYPES_ARRAY

`mbus_gem_decoder.conversion.utils.mbustypes.UNIT_TYPES_ARRAY = list(map(lambda item: item["num"], UNIT_TYPES))`

Definition at line 148 of file mbustypes.py.

6.9 mbus_gem_decoder.mbusdecode Namespace Reference

Classes

- class MBusDecode

  MBUS-GEM class for decoding data from MBUS-GEM gateway's registers.

7 Class Documentation

7.1 mbus_gem_decoder.mbusdecode.MBusDecode Class Reference

MBUS-GEM class for decoding data from MBUS-GEM gateway's registers.

Public Member Functions

- None __init__ (self, list[int] ten_regs, int gw_reg, int human=0)

  Constructor.
- str __str__ (self)
- object to_object (self)

  Convert class to object.

Public Attributes

- ten_regs
- gw_reg
- human
- conversion

7.1.1 Detailed Description

MBUS-GEM class for decoding data from MBUS-GEM gateway's registers.

Definition at line 28 of file mbusdecode.py.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 __init__() None mbus_gem_decoder.mbusdecode.MBusDecode.__init__ (

```
 self,
list[int] ten_regs,
int gw_reg,
int  human = 0 )
```

Constructor.

Parameters

ten_regs	Ten register values as list of integers.
gw_reg	Register as declared in the MBUS-GEM gateway. human (int, optional): Generate human readable values: 0 – ignore,
1	only human readable values, 2 – both. Defaults to 0.

Definition at line 38 of file mbusdecode.py.

```
00042         """
00043
00044         if gw_reg < 0:
00045             raise Exception("Gateway register cannot be negative")
00046         if len(ten_regs) != 10:
00047             raise Exception("Must provide exactly ten register values")
00048         # if human not in [0, 1, 2]:
00049         if human not in READABILITY_TYPES_ARRAY:
00050             raise Exception("Illegal value for human readability")
00051         self.ten_regs = ten_regs
00052         self.gw_reg = gw_reg
00053         self.human = human
00054         self.conversion = None
00055         reg_type = abs(self.ten_regs[7] » 8)
00056         if reg_type in REGISTER_TYPES_ARRAY:
00057             if reg_type == RegisterType.METER_ENTRY.value:
00058                 self.conversion = MBusMeterEntry(
00059                     self.ten_regs, self.gw_reg, self.human)
00060             elif reg_type == RegisterType.METER.value:
00061                 self.conversion = MBusMeter(
00062                     self.ten_regs, self.gw_reg, self.human)
00063             elif reg_type == RegisterType.MBUS_GEM.value:
00064                 self.conversion = MBusMBus(
00065                     self.ten_regs, self.gw_reg, self.human)
00066             else:
00067                 raise Exception("Illegal register type")
00068         else:
00069             raise Exception("Illegal register type")
00070
00071             # self.conversion = switch_.get(
00072             #     RegisterType(abs(self.ten_regs[7] » 8)), {})
00073
```

7.1.3 Member Function Documentation

7.1.3.1 __str__() `str mbus_gem_decoder.mbusdecode.MBusDecode.__str__ ( self )`

Definition at line 74 of file mbusdecode.py.

```
00074     def __str__(self) -> str:
00075         if isinstance(self.conversion, type(None)):
00076             raise Exception("Conversion failed")
00077         return str(self.conversion)
00078
```

7.1.3.2 to_object() `object mbus_gem_decoder.mbusdecode.MBusDecode.to_object ( self )`

Convert class to object.

Exceptions

Exception	Conversion has failed

Returns

object Converted object

Definition at line 87 of file mbusdecode.py.

```
00087         """
00088
00089         if isinstance(self.conversion, type(None)):
00090             raise Exception("Conversion failed")
00091         return self.conversion.to_object()
```

7.1.4 Member Data Documentation

7.1.4.1 conversion `mbus_gem_decoder.mbusdecode.MBusDecode.conversion`

Definition at line 54 of file mbusdecode.py.

7.1.4.2 gw_reg `mbus_gem_decoder.mbusdecode.MBusDecode.gw_reg`

Definition at line 52 of file mbusdecode.py.

7.1.4.3 human `mbus_gem_decoder.mbusdecode.MBusDecode.human`

Definition at line 53 of file mbusdecode.py.

7.1.4.4 ten_regs `mbus_gem_decoder.mbusdecode.MBusDecode.ten_regs`

Definition at line 51 of file mbusdecode.py.

The documentation for this class was generated from the following file:

- mbus_gem_decoder/mbusdecode.py

7.2 mbus_gem_decoder.conversion.mbusmbus.MBusMBus Class Reference

MBUS-GEM gateway class.

Public Member Functions

- None __init__ (self, list[int] ten_regs, int gw_reg, int human=0)

  Constructor.
- object convert_data_in_regs_mbusgem (self)

  Convert registers that hold data about the MBUS-GEM gateway.
- object to_object (self)

  Convert to object.
- def __str__ (self)

Public Attributes

- ten_regs
- gw_reg
- human
- reg
- serial
- protocol
- version
- unix_timestamp
- register_type

7.2.1 Detailed Description

MBUS-GEM gateway class.

Definition at line 16 of file mbusmbus.py.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 __init__() None mbus_gem_decoder.conversion.mbusmbus.MBusMBus.__init__ (self, list[int] ten_regs, int gw_reg, int human = 0)

Constructor.

Parameters

ten_regs	Ten register values as list of integers.
gw_reg	Register as declared in the MBUS-GEM gateway. human (int, optional): Generate human readable values: 0 – ignore,
1	only human readable values, 2 – both. Defaults to 0.

Definition at line 25 of file mbusmbus.py.

```
00029         """
00030
00031         if gw_reg < 0:
00032             raise Exception("Gateway register cannot be negative")
00033         if len(ten_regs) != 10:
00034             raise Exception("Must provide exactly ten register values")
00035         # if human not in [0, 1, 2]:
00036         if human not in READABILITY_TYPES_ARRAY:
00037             raise Exception("Illegal value for human readability")
00038         self.ten_regs = ten_regs
00039         self.gw_reg = gw_reg
00040         self.human = human
00041         self.convert_data_in_regs_mbusgem()
00042
```

7.2.3 Member Function Documentation

7.2.3.1 __str__() `def mbus_gem_decoder.conversion.mbusmbus.MBusMBus.__str__ ( self )`

Definition at line 90 of file mbusmbus.py.

```
00090     def __str__(self):
00091         data = self.to_object()
00092         return json.dumps(data)
```

7.2.3.2 convert_data_in_regs_mbusgem() `object mbus_gem_decoder.conversion.mbusmbus.MBusMBus.convert_data_in_regs_mbusgem ( self )`

Convert registers that hold data about the MBUS-GEM gateway.

Returns

object Object with human-readable data

REG	VALUE	SIZE	DETAILS
0-1	Serial number	32bit	Serial number of MBUS-GEM as hexadecimal
			number
2	Protocol version	16bit	Protocol version for ModBus interface
			(value=1)
3	Version	16bit	Software version of the gateway (as
			integer)
4-5	Time stamp	32bit	Unix time stamp of last read-out
6	Reserved	16bit	
7	Type field	16bit	Type field for register set in the upper
			byte, lower byte is reserved
8-9	Reserved	32bit	

Definition at line 62 of file mbusmbus.py.

```
00062         """
00063
00064         self.reg = self.gw_reg
```

```
00065         self.serial = mbus_serial(self.ten_regs)
00066         self.protocol = mbus_protocol_version(self.ten_regs)
00067         self.version = mbus_version(self.ten_regs)
00068         self.unix_timestamp = get_unix_timestamp(self.ten_regs)
00069         self.register_type = register_type(self.ten_regs)
00070
```

7.2.3.3 to_object() object mbus_gem_decoder.conversion.mbusmbus.MBusMBus.to_object (self)

Convert to object.

Returns

object data as object

Definition at line 76 of file mbusmbus.py.

```
00076         """
00077         data = {}
00078         data["reg"] = self.reg
00079         data["serialNo"] = self.serial
00080         data["protocolVersion"] = self.protocol
00081         data["version"] = self.version
00082         if self.human in [0, 2]:
00083             data["timestampUnix"] = self.unix_timestamp
00084             data["type"] = self.register_type
00085         if self.human in [1, 2]:
00086             data["timestamp"] = get_timestamp(self.ten_regs)
00087             data["type_string"] = register_type_str(self.ten_regs)
00088         return data
00089
```

7.2.4 Member Data Documentation

7.2.4.1 gw_reg mbus_gem_decoder.conversion.mbusmbus.MBusMBus.gw_reg

Definition at line 39 of file mbusmbus.py.

7.2.4.2 human mbus_gem_decoder.conversion.mbusmbus.MBusMBus.human

Definition at line 40 of file mbusmbus.py.

7.2.4.3 protocol mbus_gem_decoder.conversion.mbusmbus.MBusMBus.protocol

Definition at line 66 of file mbusmbus.py.

7.2.4.4 reg `mbus_gem_decoder.conversion.mbusmbus.MBusMBus.reg`

Definition at line 64 of file mbusmbus.py.

7.2.4.5 register_type `mbus_gem_decoder.conversion.mbusmbus.MBusMBus.register_type`

Definition at line 69 of file mbusmbus.py.

7.2.4.6 serial `mbus_gem_decoder.conversion.mbusmbus.MBusMBus.serial`

Definition at line 65 of file mbusmbus.py.

7.2.4.7 ten_regs `mbus_gem_decoder.conversion.mbusmbus.MBusMBus.ten_regs`

Definition at line 38 of file mbusmbus.py.

7.2.4.8 unix_timestamp `mbus_gem_decoder.conversion.mbusmbus.MBusMBus.unix_timestamp`

Definition at line 68 of file mbusmbus.py.

7.2.4.9 version `mbus_gem_decoder.conversion.mbusmbus.MBusMBus.version`

Definition at line 67 of file mbusmbus.py.

The documentation for this class was generated from the following file:

- mbus_gem_decoder/conversion/mbusmbus.py

7.3 mbus_gem_decoder.conversion.mbusmeter.MBusMeter Class Reference

MBUS-GEM METER class.

Public Member Functions

- None __init__ (self, list[int] ten_regs, int gw_reg, int human=0)
  Constructor.
- object convert_data_in_regsmeter (self)
  Convert registers that hold data about a METER.
- object to_object (self)
  Convert to object.
- def __str__ (self)

Public Attributes

- ten_regs
- gw_reg
- human
- reg
- serial
- manufacturer
- version
- medium
- unix_timestamp
- flag1
- flag2
- register_type

7.3.1 Detailed Description

MBUS-GEM METER class.

Definition at line 17 of file mbusmeter.py.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 __init__() None mbus_gem_decoder.conversion.mbusmeter.MBusMeter.__init__ (self, list[int] ten_regs, int gw_reg, int human = 0)

Constructor.

Parameters

ten_regs	Ten register values as list of integers.
gw_reg	Register as declared in the MBUS-GEM gateway. human (int, optional): Generate human readable values: 0 – ignore,
1	only human readable values, 2 – both. Defaults to 0.

Definition at line 26 of file mbusmeter.py.

```
00030         """
00031
00032         if gw_reg < 0:
00033             raise Exception("Gateway register cannot be negative")
00034         if len(ten_regs) != 10:
00035             raise Exception("Must provide exactly ten register values")
00036         # if human not in [0, 1, 2]:
00037         if human not in READABILITY_TYPES_ARRAY:
00038             raise Exception("Illegal value for human readability")
00039         self.ten_regs = ten_regs
00040         self.gw_reg = gw_reg
00041         self.human = human
00042         self.convert_data_in_regsmeter()
00043
```

7.3.3 Member Function Documentation

7.3.3.1 __str__() def mbus_gem_decoder.conversion.mbusmeter.MBusMeter.__str__ (self)

Definition at line 105 of file mbusmeter.py.

```
00105     def __str__(self):
00106         data = self.to_object()
00107         return json.dumps(data)
```

7.3.3.2 convert_data_in_regsmeter() object mbus_gem_decoder.conversion.mbusmeter.MBusMeter.convert_data_in_regsmeter (self)

Convert registers that hold data about a METER.

Returns

object Object with human-readable data

REG	VALUE	SIZE	DETAILS
0-1	Serial number	32bit	Serial number of meter as integer value
2	Manufacturer ID	16bit	Encoding of manufacturer by using
			different blocks of bits:
			10-14, 1st; 5-9, 2nd; 0-4, 3rd (A=1)
3	Version/Medium	16bit	Version of meter: upper byte; Medium:
			lower byte
4-5	Time stamp	32bit	Unix time stamp of last read-out
6	Reserved	16bit	
7	Type field	16bit	Type field for register set in the upper
			byte, lower byte is reserved
8	Flags/Reserved	16bit	bit[0]=1: meter could not be read;
			bit[0]=0: could be read correctly;
			bit[1]=1: not all values are updated;
			bit[1]=0: all meter values updated;
			bit[2:15] reserved;
9	Reserved	16bit	

Definition at line 68 of file mbusmeter.py.

```
00068         """
00069
00070         self.reg = self.gw_reg
00071         self.serial = meter_serial(self.ten_regs)
00072         self.manufacturer = meter_manufacturer(self.ten_regs)
00073         self.version = meter_version(self.ten_regs)
00074         self.medium = meter_medium(self.ten_regs)
00075         self.unix_timestamp = get_unix_timestamp(self.ten_regs)
00076         self.flag1 = meter_flag1(self.ten_regs)
00077         self.flag2 = meter_flag2(self.ten_regs)
00078         self.register_type = register_type(self.ten_regs)
00079
```

7.3.3.3 to_object() object mbus_gem_decoder.conversion.mbusmeter.MBusMeter.to_object (self)

Convert to object.

Returns

object data as object

Definition at line 85 of file mbusmeter.py.

```
00085         """
00086         data = {}
00087         data["reg"] = self.reg
00088         data["serialNo"] = self.serial
00089         data["manufacturerID"] = self.manufacturer
00090         data["version"] = self.version
00091         if self.human in [0, 2]:
00092             data["medium"] = self.medium
00093             data["timestampUnix"] = self.unix_timestamp
00094             data["flag1"] = self.flag1
00095             data["flag2"] = self.flag2
00096             data["type"] = self.register_type
00097         if self.human in [1, 2]:
00098             data["medium_string"] = meter_medium_str(self.medium)
00099             data["timestamp"] = get_timestamp(self.ten_regs)
00100             data["flag1_string"] = meter_flag1_str(self.ten_regs)
00101             data["flag2_string"] = meter_flag2_str(self.ten_regs)
00102             data["type_string"] = register_type_str(self.ten_regs)
00103         return data
00104
```

7.3.4 Member Data Documentation

7.3.4.1 flag1 mbus_gem_decoder.conversion.mbusmeter.MBusMeter.flag1

Definition at line 76 of file mbusmeter.py.

7.3.4.2 flag2 mbus_gem_decoder.conversion.mbusmeter.MBusMeter.flag2

Definition at line 77 of file mbusmeter.py.

7.3.4.3 gw_reg mbus_gem_decoder.conversion.mbusmeter.MBusMeter.gw_reg

Definition at line 40 of file mbusmeter.py.

7.3.4.4 human `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.human`

Definition at line 41 of file mbusmeter.py.

7.3.4.5 manufacturer `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.manufacturer`

Definition at line 72 of file mbusmeter.py.

7.3.4.6 medium `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.medium`

Definition at line 74 of file mbusmeter.py.

7.3.4.7 reg `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.reg`

Definition at line 70 of file mbusmeter.py.

7.3.4.8 register_type `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.register_type`

Definition at line 78 of file mbusmeter.py.

7.3.4.9 serial `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.serial`

Definition at line 71 of file mbusmeter.py.

7.3.4.10 ten_regs `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.ten_regs`

Definition at line 39 of file mbusmeter.py.

7.3.4.11 unix_timestamp `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.unix_timestamp`

Definition at line 75 of file mbusmeter.py.

7.3.4.12 version `mbus_gem_decoder.conversion.mbusmeter.MBusMeter.version`

Definition at line 73 of file mbusmeter.py.

The documentation for this class was generated from the following file:

- mbus_gem_decoder/conversion/mbusmeter.py

7.4 mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry Class Reference

MBUS-GEM METER ENTRY class.

Public Member Functions

- None __init__ (self, list[int] ten_regs, int gw_reg, int human=0)
  Constructor.
- object convert_data_in_regsmeter_entry (self)
  Convert registers that hold data about a METER ENTRY.
- object to_object (self)
  Convert to object.
- def __str__ (self)

Public Attributes

- ten_regs
- gw_reg
- human
- reg
- integer
- scale
- float
- unit
- unix_timestamp
- register_type

7.4.1 Detailed Description

MBUS-GEM METER ENTRY class.

Definition at line 16 of file mbusmeterentry.py.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 __init__() `None mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.__init__ (`

```
self,
list[int] ten_regs,
int gw_reg,
int  human = 0 )
```

Constructor.

Parameters

ten_regs	Ten register values as list of integers.
gw_reg	Register as declared in the MBUS-GEM gateway. human (int, optional): Generate human readable values: 0 – ignore,
1	only human readable values, 2 – both. Defaults to 0.

Definition at line 25 of file mbusmeterentry.py.

```
00029         """
00030
00031         if gw_reg < 0:
00032             raise Exception("Gateway register cannot be negative")
00033         if len(ten_regs) != 10:
00034             raise Exception("Must provide exactly ten register values")
00035         # if human not in [0, 1, 2]:
00036         if human not in READABILITY_TYPES_ARRAY:
00037             raise Exception("Illegal value for human readability")
00038         self.ten_regs = ten_regs
00039         self.gw_reg = gw_reg
00040         self.human = human
00041         self.convert_data_in_regsmeter_entry()
00042
```

7.4.3 Member Function Documentation

7.4.3.1 __str__() def mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.__str__ (self)

Definition at line 92 of file mbusmeterentry.py.

```
00092     def __str__(self):
00093         data = self.to_object()
00094         return json.dumps(data)
```

7.4.3.2 convert_data_in_regsmeter_entry() object mbus_gem_decoder.conversion.mbusmeterentry.↩ MBusMeterEntry.convert_data_in_regsmeter_entry (self)

Convert registers that hold data about a METER ENTRY.

Get data/info about specific value of meter.

Returns

object Object of human-readable data

REG	VALUE	SIZE	DETAILS
0-3	Meter value	64bit	Signed integer (not scaled)
4-5	Meter value	32bit	Floating point value (scaled)
6	Scale factor	16bit	Signed scale factor (power of 10)
7	Type/Unit	16bit	Type field set in the upper byte; Unit
			in the lower byte
8-9	Time stamp	32bit	Unix time stamp

Definition at line 59 of file mbusmeterentry.py.

```
00059         """
00060
00061         self.reg = self.gw_reg
00062         self.integer = get_integer_value(self.ten_regs)
00063         self.scale = get_scale(self.ten_regs)
00064         self.float = get_float_value(self.ten_regs)
00065         self.unit = get_unit(self.ten_regs)
00066         self.unix_timestamp = get_unix_timestamp(self.ten_regs)
00067         self.register_type = register_type(self.ten_regs)
00068
```

7.4.3.3 to_object() object mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.to_object (

self)

Convert to object.

Returns

object data as object

Definition at line 74 of file mbusmeterentry.py.

```
00074         """
00075         data = {}
00076         data["reg"] = self.reg
00077         data["integer"] = self.integer
00078         data["scale"] = self.scale
00079         data["float"] = self.float
00080         if self.human in [0, 2]:
00081             data["unit"] = self.unit
00082             data["timestampUnix"] = self.unix_timestamp
00083             data["type"] = self.register_type
00084         if self.human in [1, 2]:
00085             unit_readable = get_unit_type(self.unit)
00086             data["unit_string"] = unit_readable["name"]
00087             data["unit_description"] = unit_readable["desc"]
00088             data["timestamp"] = get_timestamp(self.ten_regs)
00089             data["type_string"] = register_type_str(self.ten_regs)
00090         return data
00091
```

7.4.4 Member Data Documentation

7.4.4.1 float mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.float

Definition at line 64 of file mbusmeterentry.py.

7.4.4.2 gw_reg mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.gw_reg

Definition at line 39 of file mbusmeterentry.py.

7.4.4.3 human `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.human`

Definition at line 40 of file mbusmeterentry.py.

7.4.4.4 integer `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.integer`

Definition at line 62 of file mbusmeterentry.py.

7.4.4.5 reg `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.reg`

Definition at line 61 of file mbusmeterentry.py.

7.4.4.6 register_type `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.register_type`

Definition at line 67 of file mbusmeterentry.py.

7.4.4.7 scale `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.scale`

Definition at line 63 of file mbusmeterentry.py.

7.4.4.8 ten_regs `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.ten_regs`

Definition at line 38 of file mbusmeterentry.py.

7.4.4.9 unit `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.unit`

Definition at line 65 of file mbusmeterentry.py.

7.4.4.10 unix_timestamp `mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry.unix_↩timestamp`

Definition at line 66 of file mbusmeterentry.py.

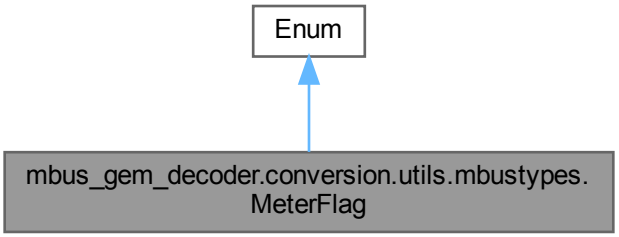
The documentation for this class was generated from the following file:

- mbus_gem_decoder/conversion/mbusmeterentry.py

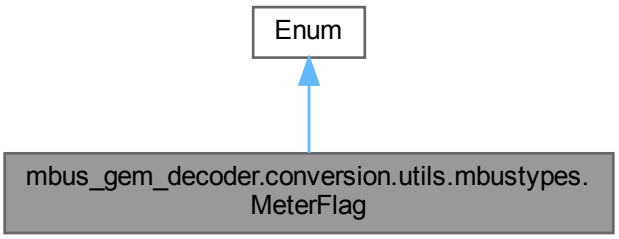
7.5 mbus_gem_decoder.conversion.utils.mbustypes.MeterFlag Class Reference

Enumerator for meter status messages.

Inheritance diagram for mbus_gem_decoder.conversion.utils.mbustypes.MeterFlag:



Collaboration diagram for mbus_gem_decoder.conversion.utils.mbustypes.MeterFlag:



Static Public Attributes

- int READ_SUCCESS = 0
- int READ_FAILURE = 1
- int VALUES_UPDATED_SUCCESS = 2
- int VALUES_UPDATE_FAILURE = 3

7.5.1 Detailed Description

Enumerator for meter status messages.

KEY	VALUE
READ_SUCCESS	0
READ_FAILURE	1
VALUES_UPDATED_SUCCESS	2
VALUES_UPDATE_FAILURE	3

Definition at line 36 of file mbustypes.py.

7.5.2 Member Data Documentation

7.5.2.1 READ_FAILURE `int mbus_gem_decoder.conversion.utils.mbustypes.MeterFlag.READ_FAILURE = 1 [static]`

Definition at line 39 of file mbustypes.py.

7.5.2.2 READ_SUCCESS `int mbus_gem_decoder.conversion.utils.mbustypes.MeterFlag.READ_SUCCESS = 0 [static]`

Definition at line 38 of file mbustypes.py.

7.5.2.3 VALUES_UPDATE_FAILURE `int mbus_gem_decoder.conversion.utils.mbustypes.MeterFlag.↩ VALUES_UPDATE_FAILURE = 3 [static]`

Definition at line 41 of file mbustypes.py.

7.5.2.4 VALUES_UPDATED_SUCCESS `int mbus_gem_decoder.conversion.utils.mbustypes.Meter↩ Flag.VALUES_UPDATED_SUCCESS = 2 [static]`

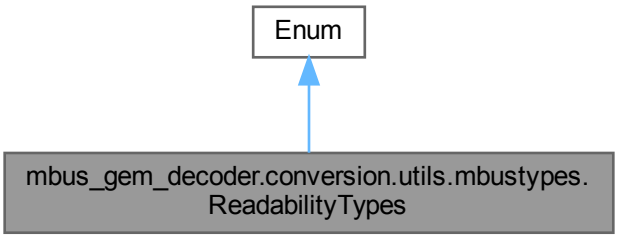
Definition at line 40 of file mbustypes.py.

The documentation for this class was generated from the following file:

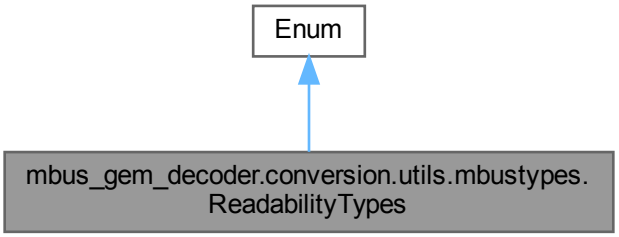
- mbus_gem_decoder/conversion/utils/mbustypes.py

7.6 mbus_gem_decoder.conversion.utils.mbustypes.ReadabilityTypes Class Reference

Inheritance diagram for mbus_gem_decoder.conversion.utils.mbustypes.ReadabilityTypes:



Collaboration diagram for mbus_gem_decoder.conversion.utils.mbustypes.ReadabilityTypes:



Static Public Attributes

- int SHORT = 0
- int HUMAN_READABLE = 1
- int BOTH = 2

7.6.1 Detailed Description

Definition at line 51 of file mbustypes.py.

7.6.2 Member Data Documentation

7.6.2.1 BOTH `int mbus_gem_decoder.conversion.utils.mbustypes.ReadabilityTypes.BOTH = 2 [static]`

Definition at line 55 of file mbustypes.py.

7.6.2.2 HUMAN_READABLE `int mbus_gem_decoder.conversion.utils.mbustypes.ReadabilityTypes.↩ HUMAN_READABLE = 1 [static]`

Definition at line 54 of file mbustypes.py.

7.6.2.3 SHORT `int mbus_gem_decoder.conversion.utils.mbustypes.ReadabilityTypes.SHORT = 0 [static]`

Definition at line 53 of file mbustypes.py.

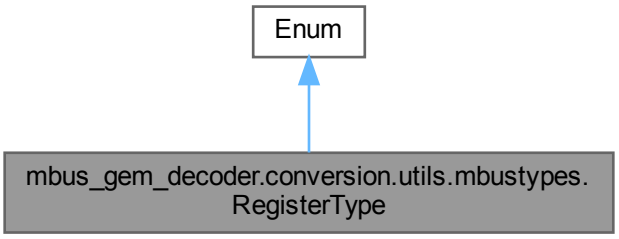
The documentation for this class was generated from the following file:

- mbus_gem_decoder/conversion/utils/mbustypes.py

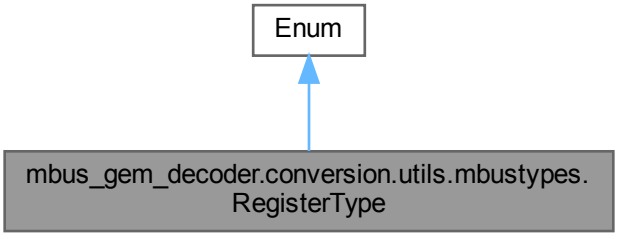
7.7 mbus_gem_decoder.conversion.utils.mbustypes.RegisterType Class Reference

Enumerator for register types.

Inheritance diagram for mbus_gem_decoder.conversion.utils.mbustypes.RegisterType:



Collaboration diagram for mbus_gem_decoder.conversion.utils.mbustypes.RegisterType:



Static Public Attributes

- int METER_ENTRY = 0
- int MBUS_GEM = 1
- int METER = 2

7.7.1 Detailed Description

Enumerator for register types.

KEY	VALUE
METER_ENTRY	0
MBUS_GEM	1
METER	2

Definition at line 19 of file mbustypes.py.

7.7.2 Member Data Documentation

7.7.2.1 MBUS_GEM `int mbus_gem_decoder.conversion.utils.mbustypes.RegisterType.MBUS_GEM = 1 [static]`

Definition at line 22 of file mbustypes.py.

7.7.2.2 METER `int mbus_gem_decoder.conversion.utils.mbustypes.RegisterType.METER = 2 [static]`

Definition at line 23 of file mbustypes.py.

7.7.2.3 METER_ENTRY `int mbus_gem_decoder.conversion.utils.mbustypes.RegisterType.METER_↩ ENTRY = 0 [static]`

Definition at line 21 of file mbustypes.py.

The documentation for this class was generated from the following file:

- mbus_gem_decoder/conversion/utils/mbustypes.py

8 File Documentation

8.1 mbus_gem_decoder/__init__.py File Reference

Namespaces

- namespace mbus_gem_decoder

8.2 __init__.py

Go to the documentation of this file.

```
00001 from .conversion.utils.mbustypes import REGISTER_TYPES_ARRAY
00002 from .conversion.utils.mbustypes import READABILITY_TYPES_ARRAY
00003 from .conversion.utils.mbustypes import RegisterType
00004 from .conversion.mbusmeter import MBusMeter
00005 from .conversion.mbusmeterentry import MBusMeterEntry
00006 from .conversion.mbusmbus import MBusMBus
00007 from .mbusdecode import MBusDecode
```

8.3 mbus_gem_decoder/conversion/__init__.py File Reference

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.conversion

8.4 __init__.py

Go to the documentation of this file.

```
00001 from .utils.mbustypes import REGISTER_TYPES_ARRAY, READABILITY_TYPES_ARRAY
00002 from .utils.helpers import *
```

8.5 mbus_gem_decoder/conversion/utils/__init__.py File Reference

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.conversion
- namespace mbus_gem_decoder.conversion.utils

8.6 __init__.py

Go to the documentation of this file.

8.7 mbus_gem_decoder/conversion/mbusmbus.py File Reference

Classes

- class mbus_gem_decoder.conversion.mbusmbus.MBusMBus

  MBUS-GEM gateway class.

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.conversion
- namespace mbus_gem_decoder.conversion.mbusmbus

8.8 mbusmbus.py

Go to the documentation of this file.

```
00001 """Convert raw data to MBUS-GEM gateway type
00002 """
00003
00004 __version__ = "0.1.0"
00005 __author__ = "René"
00006
00007 import json
00008 from . import READABILITY_TYPES_ARRAY
00009 from . import mbus_serial, mbus_protocol_version, mbus_version
00010 from . import register_type, register_type_str
00011 from . import get_unix_timestamp, get_timestamp
00012
00013
00014 class MBusMBus:
00015     """MBUS-GEM gateway class
00016     """
00017     def __init__(
00018                 self,
00019                 ten_regs: list[int],
00020                 gw_reg: int,
00021                 human: int = 0) -> None:
00022         """Constructor
00023
00024         Args:
00025             ten_regs (list[int]): Ten register values as list of integers.
00026             gw_reg (int): Register as declared in the MBUS-GEM gateway.
00027             human (int, optional): Generate human readable values: 0 -- ignore,
00028             1 -- only human readable values, 2 -- both. Defaults to 0.
00029         """
00030
00031         if gw_reg < 0:
00032             raise Exception("Gateway register cannot be negative")
00033         if len(ten_regs) != 10:
00034             raise Exception("Must provide exactly ten register values")
00035         # if human not in [0, 1, 2]:
00036         if human not in READABILITY_TYPES_ARRAY:
00037             raise Exception("Illegal value for human readability")
00038         self.ten_regs = ten_regs
00039         self.gw_reg = gw_reg
00040         self.human = human
00041         self.convert_data_in_regs_mbusgem()
00042
00043     def convert_data_in_regs_mbusgem(self) -> object:
00044         """Convert registers that hold data about the MBUS-GEM gateway.
00045
00046         Returns:
00047             object: Object with human-readable data
00048
00049         |REG| VALUE          |SIZE  | DETAILS                                  |
00050         |:--|:---------------|-----:|:-----------------------------------------|
00051         |0-1|Serial number   | 32bit| Serial number of MBUS-GEM as hexadecimal|
00052         |   |                |      | number                                   |
00053         |2  |Protocol version| 16bit| Protocol version for ModBus interface    |
00054         |   |                |      | (value=1)                                |
00055         |3  |Version         | 16bit| Software version of the gateway (as      |
00056         |   |                |      | integer)                                 |
00057         |4-5|Time stamp      | 32bit| Unix time stamp of last read-out         |
00058         |6  |Reserved        | 16bit|                                          |
00059         |7  |Type field      | 16bit| Type field for register set in the upper|
00060         |   |                |      | byte, lower byte is reserved             |
00061         |8-9| Reserved       | 32bit|                                          |
00062         """
00063
00064         self.reg = self.gw_reg
00065         self.serial = mbus_serial(self.ten_regs)
00066         self.protocol = mbus_protocol_version(self.ten_regs)
00067         self.version = mbus_version(self.ten_regs)
00068         self.unix_timestamp = get_unix_timestamp(self.ten_regs)
00069         self.register_type = register_type(self.ten_regs)
00070
00071     def to_object(self) -> object:
00072         """Convert to object
00073
00074         Returns:
00075             object: data as object
00076         """
00077         data = {}
00078         data["reg"] = self.reg
00079         data["serialNo"] = self.serial
00080         data["protocolVersion"] = self.protocol
00081         data["version"] = self.version
00082         if self.human in [0, 2]:
00083             data["timestampUnix"] = self.unix_timestamp
```

```
00084             data["type"] = self.register_type
00085         if self.human in [1, 2]:
00086             data["timestamp"] = get_timestamp(self.ten_regs)
00087             data["type_string"] = register_type_str(self.ten_regs)
00088         return data
00089
00090     def __str__(self):
00091         data = self.to_object()
00092         return json.dumps(data)
```

8.9 mbus_gem_decoder/conversion/mbusmeter.py File Reference

Classes

- class mbus_gem_decoder.conversion.mbusmeter.MBusMeter

  MBUS-GEM METER class.

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.conversion
- namespace mbus_gem_decoder.conversion.mbusmeter

8.10 mbusmeter.py

Go to the documentation of this file.

```
00001 """Convert raw data to METER type
00002 """
00003
00004 __version__ = "0.1.0"
00005 __author__ = "René"
00006
00007 import json
00008 from . import READABILITY_TYPES_ARRAY
00009 from . import meter_serial, meter_manufacturer, meter_version, meter_medium
00010 from . import meter_medium_str, meter_flag1, meter_flag2, meter_flag1_str
00011 from . import meter_flag2_str, register_type, register_type_str
00012 from . import get_unix_timestamp, get_timestamp
00013
00014
00015 class MBusMeter:
00016     """MBUS-GEM METER class
00017     """
00018     def __init__(
00019                 self,
00020                 ten_regs: list[int],
00021                 gw_reg: int,
00022                 human: int = 0) -> None:
00023         """Constructor
00024
00025         Args:
00026             ten_regs (list[int]): Ten register values as list of integers.
00027             gw_reg (int): Register as declared in the MBUS-GEM gateway.
00028             human (int, optional): Generate human readable values: 0 -- ignore,
00029             1 -- only human readable values, 2 -- both. Defaults to 0.
00030         """
00031
00032         if gw_reg < 0:
00033             raise Exception("Gateway register cannot be negative")
00034         if len(ten_regs) != 10:
00035             raise Exception("Must provide exactly ten register values")
00036         # if human not in [0, 1, 2]:
00037         if human not in READABILITY_TYPES_ARRAY:
00038             raise Exception("Illegal value for human readability")
00039         self.ten_regs = ten_regs
00040         self.gw_reg = gw_reg
00041         self.human = human
00042         self.convert_data_in_regsmeter()
00043
00044     def convert_data_in_regsmeter(self) -> object:
00045         """Convert registers that hold data about a METER.
```

```
00046
00047         Returns:
00048             object: Object with human-readable data
00049
00050         |REG| VALUE           |SIZE   | DETAILS                                  |
00051         |:--|:---------------|-----:|:-----------------------------------------|
00052         |0-1| Serial number   | 32bit| Serial number of meter as integer value |
00053         |2  | Manufacturer ID| 16bit| Encoding of manufacturer by using        |
00054         |   |                 |      | different blocks of bits:                |
00055         |   |                 |      | 10-14, 1st; 5-9, 2nd; 0-4, 3rd (A=1)     |
00056         |3  | Version/Medium  | 16bit| Version of meter: upper byte; Medium:    |
00057         |   |                 |      | lower byte                               |
00058         |4-5| Time stamp      | 32bit| Unix time stamp of last read-out         |
00059         |6  | Reserved        | 16bit|                                          |
00060         |7  | Type field      | 16bit| Type field for register set in the upper|
00061         |   |                 |      | byte, lower byte is reserved             |
00062         |8  | Flags/Reserved  | 16bit| bit[0]=1: meter could not be read;       |
00063         |   |                 |      | bit[0]=0: could be read correctly;       |
00064         |   |                 |      | bit[1]=1: not all values are updated;    |
00065         |   |                 |      | bit[1]=0: all meter values updated;      |
00066         |   |                 |      | bit[2:15] reserved;                      |
00067         |9  | Reserved        | 16bit|                                          |
00068         """
00069
00070         self.reg = self.gw_reg
00071         self.serial = meter_serial(self.ten_regs)
00072         self.manufacturer = meter_manufacturer(self.ten_regs)
00073         self.version = meter_version(self.ten_regs)
00074         self.medium = meter_medium(self.ten_regs)
00075         self.unix_timestamp = get_unix_timestamp(self.ten_regs)
00076         self.flag1 = meter_flag1(self.ten_regs)
00077         self.flag2 = meter_flag2(self.ten_regs)
00078         self.register_type = register_type(self.ten_regs)
00079
00080     def to_object(self) -> object:
00081         """Convert to object
00082
00083         Returns:
00084             object: data as object
00085         """
00086         data = {}
00087         data["reg"] = self.reg
00088         data["serialNo"] = self.serial
00089         data["manufacturerID"] = self.manufacturer
00090         data["version"] = self.version
00091         if self.human in [0, 2]:
00092             data["medium"] = self.medium
00093             data["timestampUnix"] = self.unix_timestamp
00094             data["flag1"] = self.flag1
00095             data["flag2"] = self.flag2
00096             data["type"] = self.register_type
00097         if self.human in [1, 2]:
00098             data["medium_string"] = meter_medium_str(self.medium)
00099             data["timestamp"] = get_timestamp(self.ten_regs)
00100             data["flag1_string"] = meter_flag1_str(self.ten_regs)
00101             data["flag2_string"] = meter_flag2_str(self.ten_regs)
00102             data["type_string"] = register_type_str(self.ten_regs)
00103         return data
00104
00105     def __str__(self):
00106         data = self.to_object()
00107         return json.dumps(data)
```

8.11 mbus_gem_decoder/conversion/mbusmeterentry.py File Reference

Classes

- class mbus_gem_decoder.conversion.mbusmeterentry.MBusMeterEntry

  MBUS-GEM METER ENTRY class.

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.conversion
- namespace mbus_gem_decoder.conversion.mbusmeterentry

8.12 mbusmeterentry.py

Go to the documentation of this file.

```
00001 """Convert raw data to METER ENTRY type
00002 """
00003
00004 __version__ = "0.1.0"
00005 __author__ = "René"
00006
00007 import json
00008 from . import READABILITY_TYPES_ARRAY
00009 from . import get_integer_value, get_float_value, get_scale, get_unit
00010 from . import register_type, register_type_str, get_unit_type
00011 from . import get_unix_timestamp, get_timestamp
00012
00013
00014 class MBusMeterEntry:
00015     """MBUS-GEM METER ENTRY class
00016     """
00017     def __init__(
00018                 self,
00019                 ten_regs: list[int],
00020                 gw_reg: int,
00021                 human: int = 0) -> None:
00022         """Constructor
00023
00024         Args:
00025             ten_regs (list[int]): Ten register values as list of integers.
00026             gw_reg (int): Register as declared in the MBUS-GEM gateway.
00027             human (int, optional): Generate human readable values: 0 -- ignore,
00028             1 -- only human readable values, 2 -- both. Defaults to 0.
00029         """
00030
00031         if gw_reg < 0:
00032             raise Exception("Gateway register cannot be negative")
00033         if len(ten_regs) != 10:
00034             raise Exception("Must provide exactly ten register values")
00035         # if human not in [0, 1, 2]:
00036         if human not in READABILITY_TYPES_ARRAY:
00037             raise Exception("Illegal value for human readability")
00038         self.ten_regs = ten_regs
00039         self.gw_reg = gw_reg
00040         self.human = human
00041         self.convert_data_in_regsmeter_entry()
00042
00043     def convert_data_in_regsmeter_entry(self) -> object:
00044         """Convert registers that hold data about a METER ENTRY.
00045
00046         Get data/info about specific value of meter.
00047
00048         Returns:
00049             object: Object of human-readable data
00050
00051         |REG| VALUE          |SIZE  | DETAILS                                  |
00052         |:--|:---------------|-----:|:-----------------------------------------|
00053         |0-3| Meter value    | 64bit| Signed integer (not scaled)              |
00054         |4-5| Meter value    | 32bit| Floating point value (scaled)            |
00055         |6  | Scale factor   | 16bit| Signed scale factor (power of 10)        |
00056         |7  | Type/Unit      | 16bit| Type field set in the upper byte; Unit   |
00057         |   |                |      | in the lower byte                        |
00058         |8-9| Time stamp     | 32bit| Unix time stamp                          |
00059         """
00060
00061         self.reg = self.gw_reg
00062         self.integer = get_integer_value(self.ten_regs)
00063         self.scale = get_scale(self.ten_regs)
00064         self.float = get_float_value(self.ten_regs)
00065         self.unit = get_unit(self.ten_regs)
00066         self.unix_timestamp = get_unix_timestamp(self.ten_regs)
00067         self.register_type = register_type(self.ten_regs)
00068
00069     def to_object(self) -> object:
00070         """Convert to object
00071
00072         Returns:
00073             object: data as object
00074         """
00075         data = {}
00076         data["reg"] = self.reg
00077         data["integer"] = self.integer
00078         data["scale"] = self.scale
00079         data["float"] = self.float
00080         if self.human in [0, 2]:
00081             data["unit"] = self.unit
00082             data["timestampUnix"] = self.unix_timestamp
00083             data["type"] = self.register_type
```

```
00084         if self.human in [1, 2]:
00085             unit_readable = get_unit_type(self.unit)
00086             data["unit_string"] = unit_readable["name"]
00087             data["unit_description"] = unit_readable["desc"]
00088             data["timestamp"] = get_timestamp(self.ten_regs)
00089             data["type_string"] = register_type_str(self.ten_regs)
00090         return data
00091
00092     def __str__(self):
00093         data = self.to_object()
00094         return json.dumps(data)
```

8.13 mbus_gem_decoder/conversion/utils/helpers.py File Reference

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.conversion
- namespace mbus_gem_decoder.conversion.utils
- namespace mbus_gem_decoder.conversion.utils.helpers

Functions

- str mbus_gem_decoder.conversion.utils.helpers.meter_serial (list[int] ten_regs)

  Convert METER's serial number.
- str mbus_gem_decoder.conversion.utils.helpers.meter_manufacturer (list[int] ten_regs)

  Convert a numeric manufacturer code to string/letters.
- int mbus_gem_decoder.conversion.utils.helpers.meter_version (list[int] ten_regs)

  Convert METER's version to integer value.
- int mbus_gem_decoder.conversion.utils.helpers.meter_medium (list[int] ten_regs)

  METER's medium.
- str mbus_gem_decoder.conversion.utils.helpers.meter_medium_str (int value=-1)

  Convert numeric medium type to string.
- int mbus_gem_decoder.conversion.utils.helpers.meter_flag1 (list[int] ten_regs)

  Get METER's flag1 as integer.
- str mbus_gem_decoder.conversion.utils.helpers.meter_flag1_str (list[int] ten_regs)

  Get METER's flag1 as string.
- int mbus_gem_decoder.conversion.utils.helpers.meter_flag2 (list[int] ten_regs)

  Get METER's flag2 as integer.
- str mbus_gem_decoder.conversion.utils.helpers.meter_flag2_str (list[int] ten_regs)

  Get METER's flag2 as string.
- int mbus_gem_decoder.conversion.utils.helpers.register_type (list[int] ten_regs)

  METER's type as integer.
- str mbus_gem_decoder.conversion.utils.helpers.register_type_str (list[int] ten_regs)

  METER's type as string.
- int mbus_gem_decoder.conversion.utils.helpers.get_sign_correction (list[int] ten_regs)

  Get sign correction value.
- int mbus_gem_decoder.conversion.utils.helpers.get_integer_value (list[int] ten_regs)

  Get integer value of the reading.
- int mbus_gem_decoder.conversion.utils.helpers.get_scale (list[int] ten_regs)

  Scaling factor.
- float mbus_gem_decoder.conversion.utils.helpers.get_float_value (list[int] ten_regs)

  Get floating point value.
- float mbus_gem_decoder.conversion.utils.helpers.int32_to_ieee (int val_int)

Convert Python int32 to IEEE float.

- float mbus_gem_decoder.conversion.utils.helpers.two_words_to_ieee (int val_one, int val_two)

  Convert two words (16 bit) to IEEE float (32 bit).
- int mbus_gem_decoder.conversion.utils.helpers.two_words_to_long (int val_one, int val_two)

  Convert two words (16 bit) to long (32 bit).
- int mbus_gem_decoder.conversion.utils.helpers.get_unit (list[int] ten_regs)

  Unit of measure.
- dict[str, str] mbus_gem_decoder.conversion.utils.helpers.get_unit_type (int value)

  Convert numeric unit type to string.
- str mbus_gem_decoder.conversion.utils.helpers.mbus_serial (list[int] ten_regs)

  Get serial number of MBUS-GEM gateway.
- int mbus_gem_decoder.conversion.utils.helpers.mbus_protocol_version (list[int] ten_regs)

  Protocol version.
- str mbus_gem_decoder.conversion.utils.helpers.mbus_version (list[int] ten_regs)

  Protocol version.
- int mbus_gem_decoder.conversion.utils.helpers.get_unix_timestamp (list[int] ten_regs)

  UNIX timestamp.
- str mbus_gem_decoder.conversion.utils.helpers.get_timestamp (list[int] ten_regs)

  Timestamp.

8.14 helpers.py

Go to the documentation of this file.

```
00001 """Helper methods
00002 """
00003
00004 __version__ = "0.1.0"
00005 __author__ = "René"
00006
00007
00008 import struct
00009 from datetime import datetime
00010
00011 from .mbustypes import MEDIUM_TYPES_ARRAY, MEDIUM_TYPES
00012 from .mbustypes import UNIT_TYPES_ARRAY, UNIT_TYPES
00013 from .mbustypes import REGISTER_TYPES_ARRAY, RegisterType
00014 from .mbustypes import MeterFlag
00015
00016
00017 def meter_serial(ten_regs: list[int]) -> str:
00018     """Convert METER's serial number
00019
00020     Args:
00021         ten_regs (list[int]): registers
00022
00023     Returns:
00024         str: serial number
00025     """
00026
00027     return f"{((ten_regs[0] << 16) + ten_regs[1]):08d}"
00028
00029
00030 def meter_manufacturer(ten_regs: list[int]) -> str:
00031     """Convert a numeric manufacturer code to string/letters.
00032
00033     Args:
00034         ten_regs (list[int]): registers
00035
00036     Returns:
00037         str: Three letter code of manufacturer
00038     """
00039
00040     reg = ten_regs[2]
00041     id_1 = chr(ord("A") + ((reg >> 10) & 0x1F) - 1)
00042     id_2 = chr(ord("A") + ((reg >> 5) & 0x1F) - 1)
00043     id_3 = chr(ord("A") + (reg & 0x1F) - 1)
00044     return f"{id_1}{id_2}{id_3}"
00045
00046
```

```
00047 def meter_version(ten_regs: list[int]) -> int:
00048     """Convert METER's version to integer value
00049
00050     Args:
00051         ten_regs (list[int]): registers
00052
00053     Returns:
00054         int: version number
00055     """
00056
00057     return ten_regs[3] » 8
00058
00059
00060 def meter_medium(ten_regs: list[int]) -> int:
00061     """METER's medium
00062
00063     Args:
00064         ten_regs (list[int]): registers
00065
00066     Returns:
00067         int: medium
00068     """
00069
00070     return ten_regs[3] & 0xFF
00071
00072
00073 def meter_medium_str(value: int = -1) -> str:
00074     """Convert numeric medium type to string.
00075
00076     Args:
00077         value (int): Numeric medium type. Defaults to -1.
00078
00079     Returns:
00080         str: Medium type as a string
00081     """
00082
00083     if value in MEDIUM_TYPES_ARRAY:
00084         return list(
00085             filter(
00086                 lambda x: x["num"] == value,
00087                 MEDIUM_TYPES))[0]["name"]
00088     if value < 0 or value > 255:
00089         raise Exception("Medium index is out of range")
00090     return "Reserved"
00091
00092
00093 def meter_flag1(ten_regs: list[int]) -> int:
00094     """Get METER's flag1 as integer
00095
00096     Args:
00097         ten_regs (list[int]): registers
00098
00099     Returns:
00100         int: flag1
00101     """
00102
00103     return ten_regs[8] & 0x01
00104
00105
00106 def meter_flag1_str(ten_regs: list[int]) -> str:
00107     """Get METER's flag1 as string
00108
00109     Args:
00110         ten_regs (list[int]): registers
00111
00112     Returns:
00113         str: flag1
00114     """
00115
00116     return MeterFlag(ten_regs[8] & 0x01).name
00117
00118
00119 def meter_flag2(ten_regs: list[int]) -> int:
00120     """Get METER's flag2 as integer
00121
00122     Args:
00123         ten_regs (list[int]): registers
00124
00125     Returns:
00126         int: flag2
00127     """
00128
00129     return ((ten_regs[8] » 1) & 0x01) + 2
00130
00131
00132 def meter_flag2_str(ten_regs: list[int]) -> str:
00133     """Get METER's flag2 as string
```

```
00134
00135     Args:
00136         ten_regs (list[int]): registers
00137
00138     Returns:
00139         str: flag2
00140     """
00141
00142     return MeterFlag(((ten_regs[8] » 1) & 0x01) + 2).name
00143
00144
00145 def register_type(ten_regs: list[int]) -> int:
00146     """METER's type as integer
00147
00148     Args:
00149         ten_regs (list[int]): registers
00150
00151     Returns:
00152         int: type
00153     """
00154
00155     register_type_value = abs(ten_regs[7] » 8)
00156     if register_type_value not in REGISTER_TYPES_ARRAY:
00157         raise Exception("Register type value out of range")
00158     return register_type_value
00159
00160
00161 def register_type_str(ten_regs: list[int]) -> str:
00162     """METER's type as string
00163
00164     Args:
00165         ten_regs (list[int]): registers
00166
00167     Returns:
00168         str: type
00169     """
00170
00171     register_type_value = abs(ten_regs[7] » 8)
00172     if register_type_value not in REGISTER_TYPES_ARRAY:
00173         raise Exception("Register type value out of range")
00174     return RegisterType(register_type_value).name
00175
00176
00177 def get_sign_correction(ten_regs: list[int]) -> int:
00178     """Get sign correction value
00179
00180     Args:
00181         ten_regs (list[int]): registers
00182
00183     Returns:
00184         int: sign correction  (1 or -1)
00185     """
00186
00187     return -1 if (ten_regs[0] » 15) & 0x01 else 1
00188
00189
00190 def get_integer_value(ten_regs: list[int]) -> int:
00191     """Get integer value of the reading.
00192
00193     Args:
00194         ten_regs (list[int]): List of four (4) 16-bit integer values
00195
00196     Returns:
00197         int: 64-bit integer value
00198     """
00199
00200     sign_correction = get_sign_correction(ten_regs)
00201     raw_value = (ten_regs[0] & 0x3FFF) « 16
00202     raw_value = (raw_value + ten_regs[1]) « 16
00203     raw_value = (raw_value + ten_regs[2]) « 16
00204     raw_value = raw_value + ten_regs[3]
00205     return abs(sign_correction * raw_value) * sign_correction
00206
00207
00208 def get_scale(ten_regs: list[int]) -> int:
00209     """Scaling factor
00210
00211     Args:
00212         ten_regs (list[int]): registers
00213
00214     Returns:
00215         int: scaling factor
00216
00217     \f[float \approx integer\times 10^{scale}\f]
00218     """
00219
00220     correction = -1 if (ten_regs[6] » 15) & 0x01 else 1
```

```
00221     return abs((correction * ten_regs[6]) & 0x7FFF) * correction
00222
00223
00224 def get_float_value(ten_regs: list[int]) -> float:
00225     """Get floating point value
00226
00227     Args:
00228         ten_regs (list[int]): registers
00229
00230     Returns:
00231         float: approximate floating point value
00232     """
00233
00234     return two_words_to_ieee(ten_regs[4], ten_regs[5])
00235
00236
00237 def int32_to_ieee(val_int: int) -> float:
00238     """Convert Python int32 to IEEE float.
00239
00240     Args:
00241         val_int (int): Value to convert to float
00242
00243     Returns:
00244         float: Converted value as an IEEE floating point
00245     """
00246
00247     return struct.unpack("f", struct.pack("I", val_int))[0]
00248
00249
00250 def two_words_to_ieee(val_one: int, val_two: int) -> float:
00251     """Convert two words (16 bit) to IEEE float (32 bit).
00252
00253     Args:
00254         val_one (int): The first value (16 bit)
00255         val_two (int): The second value (16 bit)
00256
00257     Returns:
00258         float: Combined IEEE float (32 bit)
00259     """
00260
00261     val_int32 = two_words_to_long(val_one, val_two)
00262     return int32_to_ieee(val_int32)
00263
00264
00265 def two_words_to_long(val_one: int, val_two: int) -> int:
00266     """Convert two words (16 bit) to long (32 bit).
00267
00268     Args:
00269         val_one (int): The first value (16 bit)
00270         val_two (int): The second value (16 bit)
00271
00272     Returns:
00273         int: Combined long (32 bit)
00274     """
00275
00276     return (val_one « 16) + val_two
00277
00278
00279 def get_unit(ten_regs: list[int]) -> int:
00280     """Unit of measure
00281
00282     Args:
00283         ten_regs (list[int]): registers
00284
00285     Returns:
00286         int: Unit of measure
00287     """
00288
00289     return ten_regs[7] & 0xFF
00290
00291
00292 def get_unit_type(value: int) -> dict[str, str]:
00293     """Convert numeric unit type to string.
00294
00295     Args:
00296         value (int): Numeric unit type
00297
00298     Returns:
00299         dict[str, str]: Unit type as a dictionary of strings
00300         (name (short form), desc (description or full form of the unit))
00301     """
00302
00303     if value in UNIT_TYPES_ARRAY:
00304         unit_match = list(
00305             filter(
00306                 lambda x: x["num"] == value, UNIT_TYPES))
00307         return list(
```

```
00308             map(
00309                 lambda x:
00310                 {
00311                     "name": x["name"],
00312                     "desc": x["desc"]
00313                 }, unit_match))[0]
00314     if value < 0 or value > 255:
00315         raise Exception("Unit value out of range")
00316     return {"name": "Res", "desc": "Reserved"}
00317
00318
00319 def mbus_serial(ten_regs: list[int]) -> str:
00320     """Get serial number of MBUS-GEM gateway
00321
00322     Args:
00323         ten_regs (list[int]): registesr
00324
00325     Returns:
00326         str: serial number
00327     """
00328
00329     return f"{ten_regs[0]:04X}{ten_regs[1]:04X}"
00330
00331
00332 def mbus_protocol_version(ten_regs: list[int]) -> int:
00333     """Protocol version
00334
00335     Args:
00336         ten_regs (list[int]): registers
00337
00338     Returns:
00339         int: Protocol version
00340     """
00341
00342     return ten_regs[2]
00343
00344
00345 def mbus_version(ten_regs: list[int]) -> str:
00346     """Protocol version
00347
00348     Args:
00349         ten_regs (list[int]): registers
00350
00351     Returns:
00352         str: Protocol version
00353     """
00354
00355     return f"{ten_regs[3]//100}.{ten_regs[3]%100}"
00356
00357
00358 def get_unix_timestamp(ten_regs: list[int]) -> int:
00359     """UNIX timestamp
00360
00361     Args:
00362         ten_regs (list[int]): registers
00363
00364     Returns:
00365         int: UNIX timestamp
00366     """
00367
00368     return (ten_regs[4] « 16) + ten_regs[5]
00369
00370
00371 def get_timestamp(ten_regs: list[int]) -> str:
00372     """Timestamp
00373
00374     Args:
00375         ten_regs (list[int]): registers
00376
00377     Returns:
00378         str: Timestamp
00379     """
00380
00381     unix_time = get_unix_timestamp(ten_regs)
00382     return datetime.utcfromtimestamp(unix_time).strftime("%Y-%m-%d %H:%M:%S")
```

8.15 mbus_gem_decoder/conversion/utils/mbustypes.py File Reference

Classes

- class mbus_gem_decoder.conversion.utils.mbustypes.RegisterType

Enumerator for register types.

- class mbus_gem_decoder.conversion.utils.mbustypes.MeterFlag

  Enumerator for meter status messages.

- class mbus_gem_decoder.conversion.utils.mbustypes.ReadabilityTypes

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.conversion
- namespace mbus_gem_decoder.conversion.utils
- namespace mbus_gem_decoder.conversion.utils.mbustypes

Variables

- list mbus_gem_decoder.conversion.utils.mbustypes.MEDIUM_TYPES
- list mbus_gem_decoder.conversion.utils.mbustypes.UNIT_TYPES
- mbus_gem_decoder.conversion.utils.mbustypes.REGISTER_TYPES_ARRAY = list(map(lambda item: item.value, RegisterType))
- mbus_gem_decoder.conversion.utils.mbustypes.UNIT_TYPES_ARRAY = list(map(lambda item: item["num"], UNIT_TYPES))
- mbus_gem_decoder.conversion.utils.mbustypes.MEDIUM_TYPES_ARRAY = list(map(lambda item: item["num"], MEDIUM_TYPES))
- mbus_gem_decoder.conversion.utils.mbustypes.METER_TYPES_ARRAY = list(map(lambda item: item.value, MeterFlag))
- mbus_gem_decoder.conversion.utils.mbustypes.READABILITY_TYPES_ARRAY = list(map(lambda item: item.value, ReadabilityTypes))

8.16 mbustypes.py

Go to the documentation of this file.

```
00001 """Type constants for MBUS-GEM data conversion
00002 """
00003
00004 __version__ = "0.1.0"
00005 __author__ = "René"
00006
00007 from enum import Enum
00008
00009
00010 class RegisterType(Enum):
00011     """
00012     Enumerator for register types.
00013
00014     |   KEY       |VALUE|
00015     |:------------|----:|
00016     | METER_ENTRY | 0   |
00017     | MBUS_GEM    | 1   |
00018     | METER       | 2   |
00019     """
00020
00021     METER_ENTRY = 0
00022     MBUS_GEM = 1
00023     METER = 2
00024
00025
00026 class MeterFlag(Enum):
00027     """
00028     Enumerator for meter status messages.
00029
00030     |   KEY                 |VALUE|
00031     |:----------------------|----:|
00032     |READ_SUCCESS           | 0   |
00033     |READ_FAILURE           | 1   |
00034     |VALUES_UPDATED_SUCCESS | 2   |
00035     |VALUES_UPDATE_FAILURE  | 3   |
```

```
00036     """
00037
00038     READ_SUCCESS = 0
00039     READ_FAILURE = 1
00040     VALUES_UPDATED_SUCCESS = 2
00041     VALUES_UPDATE_FAILURE = 3
00042
00043
00044 class ReadabilityTypes(Enum):
00045     """
00046     | KEY           |VALUE|
00047     |:--------------|----:|
00048     |SHORT          | 0   |
00049     |HUMAN_READABLE| 1   |
00050     |BOTH           | 2   |
00051     """
00052
00053     SHORT = 0
00054     HUMAN_READABLE = 1
00055     BOTH = 2
00056
00057
00058 MEDIUM_TYPES = [
00059     {"num": 0, "name": "Other"},
00060     {"num": 1, "name": "Oil"},
00061     {"num": 2, "name": "Electricity"},
00062     {"num": 3, "name": "Gas"},
00063     {"num": 4, "name": "Heat (outlet)"},
00064     {"num": 5, "name": "Steam"},
00065     {"num": 6, "name": "Warm water"},
00066     {"num": 7, "name": "Water"},
00067     {"num": 8, "name": "Heat cost allocator"},
00068     {"num": 9, "name": "Compressed air"},
00069     {"num": 10, "name": "Cooling (outlet)"},
00070     {"num": 11, "name": "Cooling (inlet)"},
00071     {"num": 12, "name": "Heat (inlet)"},
00072     {"num": 13, "name": "Combined heat/cooling"},
00073     {"num": 14, "name": "Bus/System component"},
00074     {"num": 15, "name": "Unknown medium"},
00075     {"num": 20, "name": "Calorific value"},
00076     {"num": 21, "name": "Hot water"},
00077     {"num": 22, "name": "Cold water"},
00078     {"num": 23, "name": "Dual register (hot/cold) water"},
00079     {"num": 24, "name": "Pressure"},
00080     {"num": 25, "name": "A/D Converter"},
00081     {"num": 26, "name": "Smoke detector"},
00082     {"num": 27, "name": "Room sensor"},
00083     {"num": 28, "name": "Gas detector"},
00084     {"num": 32, "name": "Breaker (electricity)"},
00085     {"num": 33, "name": "Valve (gas or water)"},
00086     {"num": 37, "name": "Customer unit"},
00087     {"num": 40, "name": "Waste water"},
00088     {"num": 41, "name": "Waste"},
00089     {"num": 42, "name": "Carbon dioxide"},
00090     {"num": 49, "name": "Communication controller"},
00091     {"num": 50, "name": "Unidirectional repeater"},
00092     {"num": 51, "name": "Bidirectional repeater"},
00093     {"num": 54, "name": "Radio converter (system side)"},
00094     {"num": 55, "name": "Radio converter (meter side)"}
00095 ]
00096
00097 UNIT_TYPES = [
00098     {"num": 0, "name": "None", "desc": "None"},
00099     {"num": 1, "name": "Bin", "desc": "Binary"},
00100     {"num": 2, "name": "Cur", "desc": "Local currency unit"},
00101     {"num": 3, "name": "V", "desc": "Volt"},
00102     {"num": 4, "name": "A", "desc": "Ampere"},
00103     {"num": 5, "name": "Wh", "desc": "Watt hour"},
00104     {"num": 6, "name": "J", "desc": "Joule"},
00105     {"num": 7, "name": "m^3", "desc": "Cubic meter"},
00106     {"num": 8, "name": "kg", "desc": "Kilogram"},
00107     {"num": 9, "name": "s", "desc": "Second"},
00108     {"num": 10, "name": "min", "desc": "Minute"},
00109     {"num": 11, "name": "h", "desc": "Hour"},
00110     {"num": 12, "name": "d", "desc": "Day"},
00111     {"num": 13, "name": "W", "desc": "Watt"},
00112     {"num": 14, "name": "J/h", "desc": "Joule per hour"},
00113     {"num": 15, "name": "m^3/h", "desc": "Cubic meter per hour"},
00114     {"num": 16, "name": "m^3/min", "desc": "Cubic meter per minute"},
00115     {"num": 17, "name": "m^3/s", "desc": "Cubic meter per second"},
00116     {"num": 18, "name": "kg/h", "desc": "Kilogram per hour"},
00117     {"num": 19, "name": "Degree C", "desc": "Degree celsius"},
00118     {"num": 20, "name": "K", "desc": "Kelvin"},
00119     {"num": 21, "name": "Bar", "desc": "Bar"},
00120     {"num": 22, "name": "", "desc": "Dimensionless"},
00121     {"num": 25, "name": "UTC", "desc": "UTC"},
00122     {"num": 26, "name": "bd", "desc": "Baud"},
```

```
00123     {"num": 27, "name": "bt", "desc": "Bit time"},
00124     {"num": 28, "name": "mon", "desc": "Month"},
00125     {"num": 29, "name": "y", "desc": "Year"},
00126     {"num": 30, "name": "", "desc": "Day of week"},
00127     {"num": 31, "name": "dBm", "desc": "dBm"},
00128     {"num": 32, "name": "Bin", "desc": "Bin"},
00129     {"num": 33, "name": "Bin", "desc": "Bin"},
00130     {"num": 34, "name": "kVARh", "desc": "Kilo voltampere reactive hour"},
00131     {"num": 35, "name": "kVAR", "desc": "Kilo voltampere reactive"},
00132     {"num": 36, "name": "cal", "desc": "Calorie"},
00133     {"num": 37, "name": "%", "desc": "Percent"},
00134     {"num": 38, "name": "ft^3", "desc": "Cubic feet"},
00135     {"num": 39, "name": "Degree", "desc": "Degree"},
00136     {"num": 40, "name": "Hz", "desc": "Hertz"},
00137     {"num": 41, "name": "kBTU", "desc": "Kilo british thermal unit"},
00138     {"num": 42, "name": "mBTU/s",
00139         "desc": "Milli british thermal unit per second"},
00140     {"num": 43, "name": "US gal", "desc": "US gallon"},
00141     {"num": 44, "name": "US gal/s", "desc": "US gallon per second"},
00142     {"num": 45, "name": "US gal/min", "desc": "US gallon per minute"},
00143     {"num": 46, "name": "US gal/h", "desc": "US gallon per hour"},
00144     {"num": 47, "name": "Degree F", "desc": "Degree Fahrenheit"}
00145 ]
00146
00147 REGISTER_TYPES_ARRAY = list(map(lambda item: item.value, RegisterType))
00148 UNIT_TYPES_ARRAY = list(map(lambda item: item["num"], UNIT_TYPES))
00149 MEDIUM_TYPES_ARRAY = list(map(lambda item: item["num"], MEDIUM_TYPES))
00150 METER_TYPES_ARRAY = list(map(lambda item: item.value, MeterFlag))
00151 READABILITY_TYPES_ARRAY = list(map(lambda item: item.value, ReadabilityTypes))
```

8.17 mbus_gem_decoder/mbusdecode.py File Reference

Classes

- class mbus_gem_decoder.mbusdecode.MBusDecode

  MBUS-GEM class for decoding data from MBUS-GEM gateway's registers.

Namespaces

- namespace mbus_gem_decoder
- namespace mbus_gem_decoder.mbusdecode

8.18 mbusdecode.py

Go to the documentation of this file.

```
00001 """MBUS-GEM
00002
00003 Package for decoding MBUS-GEM register data into JSON format, optionally in
00004 more verbose human-readable values. The decoder takes a list of ten integers
00005 and converts it to a JSON object based on the object type:
00006
00007 *MBUS-GEM (gateway)
00008 *METER
00009 *METER ENTRY
00010
00011 Author: René
00012 Version: 0.1.0
00013 Copyright: GPLv3
00014
00015 Date: 2022-11-8
00016 """
00017
00018
00019 __version__ = "0.1.0"
00020 __author__ = "René"
00021
00022 from . import RegisterType, REGISTER_TYPES_ARRAY, READABILITY_TYPES_ARRAY
00023 from . import MBusMeter, MBusMeterEntry, MBusMBus
00024
00025
00026 class MBusDecode:
```

```
00027     """MBUS-GEM class for decoding data from MBUS-GEM gateway's registers.
00028     """
00029
00030     def __init__(
00031                  self,
00032                  ten_regs: list[int],
00033                  gw_reg: int,
00034                  human: int = 0) -> None:
00035         """Constructor
00036
00037         Args:
00038             ten_regs (list[int]): Ten register values as list of integers.
00039             gw_reg (int): Register as declared in the MBUS-GEM gateway.
00040             human (int, optional): Generate human readable values: 0 -- ignore,
00041             1 -- only human readable values, 2 -- both. Defaults to 0.
00042         """
00043
00044         if gw_reg < 0:
00045             raise Exception("Gateway register cannot be negative")
00046         if len(ten_regs) != 10:
00047             raise Exception("Must provide exactly ten register values")
00048         # if human not in [0, 1, 2]:
00049         if human not in READABILITY_TYPES_ARRAY:
00050             raise Exception("Illegal value for human readability")
00051         self.ten_regs = ten_regs
00052         self.gw_reg = gw_reg
00053         self.human = human
00054         self.conversion = None
00055         reg_type = abs(self.ten_regs[7] » 8)
00056         if reg_type in REGISTER_TYPES_ARRAY:
00057             if reg_type == RegisterType.METER_ENTRY.value:
00058                 self.conversion = MBusMeterEntry(
00059                     self.ten_regs, self.gw_reg, self.human)
00060             elif reg_type == RegisterType.METER.value:
00061                 self.conversion = MBusMeter(
00062                     self.ten_regs, self.gw_reg, self.human)
00063             elif reg_type == RegisterType.MBUS_GEM.value:
00064                 self.conversion = MBusMBus(
00065                     self.ten_regs, self.gw_reg, self.human)
00066             else:
00067                 raise Exception("Illegal register type")
00068         else:
00069             raise Exception("Illegal register type")
00070
00071             # self.conversion = switch_.get(
00072             #     RegisterType(abs(self.ten_regs[7] » 8)), {})
00073
00074     def __str__(self) -> str:
00075         if isinstance(self.conversion, type(None)):
00076             raise Exception("Conversion failed")
00077         return str(self.conversion)
00078
00079     def to_object(self) -> object:
00080         """Convert class to object
00081
00082         Raises:
00083             Exception: Conversion has failed
00084
00085         Returns:
00086             object: Converted object
00087         """
00088
00089         if isinstance(self.conversion, type(None)):
00090             raise Exception("Conversion failed")
00091         return self.conversion.to_object()
```

8.19 README.md File Reference

Index