RsCmwGsmMeas

Release 3.7.30.6

Rohde & Schwarz

May 27, 2021

CONTENTS:


CMW

CHAPTER

ONE

GETTING STARTED

1.1 Introduction

RsCmwGsmMeas is a Python remote-control communication module for Rohde & Schwarz SCPI-based Test and Measurement Instruments. It represents SCPI commands as fixed APIs and hence provides SCPI autocompletion and helps you to avoid common string typing mistakes.

> Basic example of the idea:
> SCPI command:
> `SYSTem:REFerence:FREQuency:SOURce`
> Python module representation:
> writing:
> `driver.system.reference.frequency.source.set()`
> reading:
> `driver.system.reference.frequency.source.get()`

Check out this RsCmwBase example:

```
""" Example on how to use the python RsCmw auto-generated instrument driver showing:
- usage of basic properties of the cmw_base object
- basic concept of setting commands and repcaps: DISPlay:WINDow<n>:SELect
- cmw_xxx drivers reliability interface usage
"""

from RsCmwBase import *  # install from pypi.org

RsCmwBase.assert_minimum_version('3.7.90.32')
cmw_base = RsCmwBase('TCPIP::10.112.1.116::INSTR', True, False)
print(f'CMW Base IND: {cmw_base.utilities.idn_string}')
print(f'CMW Instrument options:\n{",".join(cmw_base.utilities.instrument_options)}')
cmw_base.utilities.visa_timeout = 5000

# Sends OPC after each command
cmw_base.utilities.opc_query_after_write = False
```

(continues on next page)

(continued from previous page)

```
# Checks for syst:err? after each command / query
cmw_base.utilities.instrument_status_checking = True

# DISPlay:WINDow<n>:SELect
cmw_base.display.window.select.set(repcap.Window.Win1)
cmw_base.display.window.repcap_window_set(repcap.Window.Win2)
cmw_base.display.window.select.set()

# Self-test
self_test = cmw_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmw_base.reliability.ExceptionOnError = True


# Callback to use for the reliability indicator update event
def my_reliability_handler(event_args: ReliabilityEventArgs):
        print(f'Base Reliability updated.\nContext: {event_args.context}\nMessage:
↪{event_args.message}')


# We register a callback for each change in the reliability indicator
cmw_base.reliability.on_update_handler = my_reliability_handler

# You can obtain the last value of the returned reliability
print(f"\nReliability last value: {cmw_base.reliability.last_value}, context '{cmw_base.
↪reliability.last_context}', message: {cmw_base.reliability.last_message}")

# Reference Frequency Source
cmw_base.system.reference.frequency.source_set(enums.SourceIntExt.INTernal)

# Close the session
cmw_base.close()
```

Couple of reasons why to choose this module over plain SCPI approach:

- Type-safe API using typing module
- You can still use the plain SCPI communication
- You can select which VISA to use or even not use any VISA at all
- Initialization of a new session is straight-forward, no need to set any other properties
- Many useful features are already implemented - reset, self-test, opc-synchronization, error checking, option checking
- Binary data blocks transfer in both directions
- Transfer of arrays of numbers in binary or ASCII format
- File transfers in both directions
- Events generation in case of error, sent data, received data, chunk data (in case of big data transfer)

- Multithreading session locking - you can use multiple threads talking to one instrument at the same time

1.2 Installation

RsCmwGsmMeas is hosted on pypi.org. You can install it with pip (for example, `pip.exe` for Windows), or if you are using Pycharm (and you should be :-) direct in the Pycharm `Packet Management GUI`.

Preconditions

- Installed VISA. You can skip this if you plan to use only socket LAN connection. Download the Rohde & Schwarz VISA for Windows, Linux, Mac OS from here

Option 1 - Installing with pip.exe under Windows

- Start the command console: `WinKey + R`, type `cmd` and hit ENTER
- Change the working directory to the Python installation of your choice (adjust the user name and python version in the path):

```
cd c:\Users\John\AppData\Local\Programs\Python\Python37\Scripts
```

- Install with the command: `pip install RsCmwGsmMeas`

Option 2 - Installing in Pycharm

- In Pycharm Menu `File->Settings->Project->Project Interpreter` click on the **'+'** button on the bottom left
- Type `RsCmwGsmMeas` in the search box
- If you are behind a Proxy server, configure it in the Menu: `File->Settings->Appearance->System Settings->HTTP Proxy`

For more information about Rohde & Schwarz instrument remote control, check out our Instrument_Remote_Control_Web_Series .

Option 3 - Offline Installation

If you are still reading the installation chapter, it is probably because the options above did not work for you - proxy problems, your boss saw the internet bill… Here are 5 easy step for installing the RsCmwGsmMeas offline:

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCmwGsmMeas needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCmwGsmMeas package to your computer from the pypi.org: https://pypi.org/project/RsCmwGsmMeas/#files to for example `c:\temp\`
- Start the command line `WinKey + R`, type `cmd` and hit ENTER
- Change the working directory to the Python installation of your choice (adjust the user name and python version in the path):

```
cd c:\Users\John\AppData\Local\Programs\Python\Python37\Scripts
```

- Install with the command: `pip install c:\temp\RsCmwGsmMeas-3.7.30.6.tar`

1.3 Finding Available Instruments

Like the pyvisa's ResourceManager, the RsCmwGsmMeas can search for available instruments:

```
"""
Find the instruments in your environment
"""

from RsCmwGsmMeas import *

# Use the instr_list string items as resource names in the RsCmwGsmMeas constructor
instr_list = RsCmwGsmMeas.list_resources("?*")
print(instr_list)
```

If you have more VISAs installed, the one actually used by default is defined by a secret widget called Visa Conflict Manager. You can force your program to use a VISA of your choice:

```
"""
Find the instruments in your environment with the defined VISA implementation
"""

from RsCmwGsmMeas import *

# In the optional parameter visa_select you can use for example 'rs' or 'ni'
# Rs Visa also finds any NRP-Zxx USB sensors
instr_list = RsCmwGsmMeas.list_resources('?*', 'rs')
print(instr_list)
```

Tip: We believe our R&S VISA is the best choice for our customers. Here are the reasons why:

- Small footprint
- Superior VXI-11 and HiSLIP performance
- Integrated legacy sensors NRP-Zxx support
- Additional VXI-11 and LXI devices search
- Availability for Windows, Linux, Mac OS

1.4 Initiating Instrument Session

RsCmwGsmMeas offers four different types of starting your remote-control session. We begin with the most typical case, and progress with more special ones.

Standard Session Initialization

Initiating new instrument session happens, when you instantiate the RsCmwGsmMeas object. Below, is a simple Hello World example. Different resource names are examples for different physical interfaces.

```
"""
Simple example on how to use the RsCmwGsmMeas module for remote-controlling your␣
↪instrument
Preconditions:

- Installed RsCmwGsmMeas Python module Version 3.7.30 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCmwGsmMeas import *

# A good practice is to assure that you have a certain minimum version installed
RsCmwGsmMeas.assert_minimum_version('3.7.30')
resource_string_1 = 'TCPIP::192.168.2.101::INSTR'  # Standard LAN connection (also␣
↪called VXI-11)
resource_string_2 = 'TCPIP::192.168.2.101::hislip0'  # Hi-Speed LAN connection - see␣
↪1MA208
resource_string_3 = 'GPIB::20::INSTR'  # GPIB Connection
resource_string_4 = 'USB::0x0AAD::0x0119::022019943::INSTR'  # USB-TMC (Test and␣
↪Measurement Class)

# Initializing the session
driver = RsCmwGsmMeas(resource_string_1)

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f'RsCmwGsmMeas package version: {driver.utilities.driver_version}')
print(f'Visa manufacturer: {driver.utilities.visa_manufacturer}')
print(f'Instrument full name: {driver.utilities.full_instrument_model_name}')
print(f'Instrument installed options: {",".join(driver.utilities.instrument_options)}')

# Close the session
driver.close()
```

Note: If you are wondering about the missing `ASRL1::INSTR`, yes, it works too, but come on… it's 2021.

Do not care about specialty of each session kind; RsCmwGsmMeas handles all the necessary session settings for you. You immediately have access to many identification properties in the interface `driver.utilities` . Here are same of them:

- `idn_string`
- `driver_version`
- `visa_manufacturer`
- `full_instrument_model_name`
- `instrument_serial_number`
- `instrument_firmware_version`

- `instrument_options`

The constructor also contains optional boolean arguments `id_query` and `reset`:

```
driver = RsCmwGsmMeas('TCPIP::192.168.56.101::HISLIP', id_query=True, reset=True)
```

- Setting `id_query` to True (default is True) checks, whether your instrument can be used with the RsCmwGsmMeas module.
- Setting `reset` to True (default is False) resets your instrument. It is equivalent to calling the `reset()` method.

Selecting a Specific VISA

Just like in the function `list_resources()`, the RsCmwGsmMeas allows you to choose which VISA to use:

```
"""
Choosing VISA implementation
"""

from RsCmwGsmMeas import *

# Force use of the Rs Visa. For NI Visa, use the "SelectVisa='ni'"
driver = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR', True, True, "SelectVisa='rs'")

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f"\nI am using the VISA from: {driver.utilities.visa_manufacturer}")

# Close the session
driver.close()
```

No VISA Session

We recommend using VISA when possible preferrably with HiSlip session because of its low latency. However, if you are a strict VISA denier, RsCmwGsmMeas has something for you too - **no Visa installation raw LAN socket**:

```
"""
Using RsCmwGsmMeas without VISA for LAN Raw socket communication
"""

from RsCmwGsmMeas import *

driver = RsCmwGsmMeas('TCPIP::192.168.56.101::5025::SOCKET', True, True, "SelectVisa=
↪'socket'")
print(f'Visa manufacturer: {driver.utilities.visa_manufacturer}')
print(f"\nHello, I am: '{driver.utilities.idn_string}'")

# Close the session
driver.close()
```

Warning: Not using VISA can cause problems by debugging when you want to use the communication Trace Tool. The good news is, you can easily switch to use VISA and back just by changing the constructor arguments. The rest of your code stays unchanged.

Simulating Session

If a colleague is currently occupying your instrument, leave him in peace, and open a simulating session:

```
driver = RsCmwGsmMeas('TCPIP::192.168.56.101::HISLIP', True, True, "Simulate=True")
```

More `option_string` tokens are separated by comma:

```
driver = RsCmwGsmMeas('TCPIP::192.168.56.101::HISLIP', True, True, "SelectVisa='rs',␣
↪Simulate=True")
```

Shared Session

In some scenarios, you want to have two independent objects talking to the same instrument. Rather than opening a second VISA connection, share the same one between two or more RsCmwGsmMeas objects:

```
"""
Sharing the same physical VISA session by two different RsCmwGsmMeas objects
"""

from RsCmwGsmMeas import *

driver1 = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR', True, True)
driver2 = RsCmwGsmMeas.from_existing_session(driver1)

print(f'driver1: {driver1.utilities.idn_string}')
print(f'driver2: {driver2.utilities.idn_string}')

# Closing the driver2 session does not close the driver1 session - driver1 is the
↪'session master'
driver2.close()
print(f'driver2: I am closed now')

print(f'driver1: I am  still opened and working: {driver1.utilities.idn_string}')
driver1.close()
print(f'driver1: Only now I am closed.')
```

Note: The `driver1` is the object holding the 'master' session. If you call the `driver1.close()`, the `driver2` loses its instrument session as well, and becomes pretty much useless.

1.5 Plain SCPI Communication

After you have opened the session, you can use the instrument-specific part described in the RsCmwGsmMeas API Structure. If for any reason you want to use the plain SCPI, use the `utilities` interface's two basic methods:

- `write_str()` - writing a command without an answer, for example ***RST**
- `query_str()` - querying your instrument, for example the ***IDN?** query

You may ask a question. Actually, two questions:

- **Q1**: Why there are not called `write()` and `query()` ?
- **Q2**: Where is the `read()` ?

Answer 1: Actually, there are - the `write_str()` / `write()` and `query_str()` / query() are aliases, and you can use any of them. We promote the `_str` names, to clearly show you want to work with strings. Strings in Python3 are Unicode, the *bytes* and *string* objects are not interchangeable, since one character might be represented by more than 1 byte. To avoid mixing string and binary communication, all the method names for binary transfer contain `_bin` in the name.

Answer 2: Short answer - you do not need it. Long answer - your instrument never sends unsolicited responses. If you send a set command, you use `write_str()`. For a query command, you use `query_str()`. So, you really do not need it...

Bottom line - if you are used to `write()` and `query()` methods, from pyvisa, the `write_str()` and `query_str()` are their equivalents.

Enough with the theory, let us look at an example. Simple write, and query:

```
"""
Basic string write_str / query_str
"""

from RsCmwGsmMeas import *

driver = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')
driver.utilities.write_str('*RST')
response = driver.utilities.query_str('*IDN?')
print(response)

# Close the session
driver.close()
```

This example is so-called "*University-Professor-Example*" - good to show a principle, but never used in praxis. The abovementioned commands are already a part of the driver's API. Here is another example, achieving the same goal:

```
"""
Basic string write_str / query_str
"""

from RsCmwGsmMeas import *

driver = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')
driver.utilities.reset()
print(driver.utilities.idn_string)
```

(continues on next page)

(continued from previous page)

```
# Close the session
driver.close()
```

One additional feature we need to mention here: **VISA timeout**. To simplify, VISA timeout plays a role in each `query_xxx()`, where the controller (your PC) has to prevent waiting forever for an answer from your instrument. VISA timeout defines that maximum waiting time. You can set/read it with the `visa_timeout` property:

```
# Timeout in milliseconds
driver.utilities.visa_timeout = 3000
```

After this time, the RsCmwGsmMeas raises an exception. Speaking of exceptions, an important feature of the RsCmwGsmMeas is **Instrument Status Checking**. Check out the next chapter that describes the error checking in details.

For completion, we mention other string-based `write_xxx()` and `query_xxx()` methods - all in one example. They are convenient extensions providing type-safe float/boolean/integer setting/querying features:

```
"""
Basic string write_xxx / query_xxx
"""

from RsCmwGsmMeas import *

driver = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')
driver.utilities.visa_timeout = 5000
driver.utilities.instrument_status_checking = True
driver.utilities.write_int('SWEEP:COUNT ', 10)  # sending 'SWEEP:COUNT 10'
driver.utilities.write_bool('SOURCE:RF:OUTPUT:STATE ', True)  # sending
↪'SOURCE:RF:OUTPUT:STATE ON'
driver.utilities.write_float('SOURCE:RF:FREQUENCY ', 1E9)  # sending 'SOURCE:RF:FREQUENCY␣
↪1000000000'

sc = driver.utilities.query_int('SWEEP:COUNT?')  # returning integer number sc=10
out = driver.utilities.query_bool('SOURCE:RF:OUTPUT:STATE?')  # returning boolean␣
↪out=True
freq = driver.utilities.query_float('SOURCE:RF:FREQUENCY?')  # returning float number␣
↪freq=1E9

# Close the session
driver.close()
```

Lastly, a method providing basic synchronization: `query_opc()`. It sends query ***OPC?** to your instrument. The instrument waits with the answer until all the tasks it currently has in a queue are finished. This way your program waits too, and this way it is synchronized with the actions in the instrument. Remember to have the VISA timeout set to an appropriate value to prevent the timeout exception. Here's the snippet:

```
driver.utilities.visa_timeout = 3000
driver.utilities.write_str("INIT")
driver.utilities.query_opc()

# The results are ready now to fetch
results = driver.utilities.query_str("FETCH:MEASUREMENT?")
```

Tip: Wait, there's more: you can send the ***OPC?** after each `write_xxx()` automatically:

```
# Default value after init is False
driver.utilities.opc_query_after_write = True
```

1.6 Error Checking

RsCmwGsmMeas pushes limits even further (internal R&S joke): It has a built-in mechanism that after each command/query checks the instrument's status subsystem, and raises an exception if it detects an error. For those who are already screaming: **Speed Performance Penalty!!!**, don't worry, you can disable it.

Instrument status checking is very useful since in case your command/query caused an error, you are immediately informed about it. Status checking has in most cases no practical effect on the speed performance of your program. However, if for example, you do many repetitions of short write/query sequences, it might make a difference to switch it off:

```
# Default value after init is True
driver.utilities.instrument_status_checking = False
```

To clear the instrument status subsystem of all errors, call this method:

```
driver.utilities.clear_status()
```

Instrument's status system error queue is clear-on-read. It means, if you query its content, you clear it at the same time. To query and clear list of all the current errors, use this snippet:

```
errors_list = driver.utilities.query_all_errors()
```

See the next chapter on how to react on errors.

1.7 Exception Handling

The base class for all the exceptions raised by the RsCmwGsmMeas is `RsInstrException`. Inherited exception classes:

- `ResourceError` raised in the constructor by problems with initiating the instrument, for example wrong or non-existing resource name
- `StatusException` raised if a command or a query generated error in the instrument's error queue
- `TimeoutException` raised if a visa timeout or an opc timeout is reached

In this example we show usage of all of them. Because it is difficult to generate an error using the instrument-specific SCPI API, we use plain SCPI commands:

```
"""
Showing how to deal with exceptions
"""

from RsCmwGsmMeas import *

```

(continues on next page)

(continued from previous page)

```
driver = None
# Try-catch for initialization. If an error occures, the ResourceError is raised
try:
    driver = RsCmwGsmMeas('TCPIP::10.112.1.179::HISLIP')
except ResourceError as e:
    print(e.args[0])
    print('Your instrument is probably OFF...')
    # Exit now, no point of continuing
    exit(1)

# Dealing with commands that potentially generate errors OPTION 1:
# Switching the status checking OFF termporarily
driver.utilities.instrument_status_checking = False
driver.utilities.write_str('MY:MISSpelled:COMMand')
# Clear the error queue
driver.utilities.clear_status()
# Status checking ON again
driver.utilities.instrument_status_checking = True

# Dealing with queries that potentially generate errors OPTION 2:
try:
    # You migh want to reduce the VISA timeout to avoid long waiting
    driver.utilities.visa_timeout = 1000
    driver.utilities.query_str('MY:WRONg:QUERy?')

except StatusException as e:
    # Instrument status error
    print(e.args[0])
    print('Nothing to see here, moving on...')

except TimeoutException as e:
    # Timeout error
    print(e.args[0])
    print('That took a long time...')

except RsInstrException as e:
    # RsInstrException is a base class for all the RsCmwGsmMeas exceptions
    print(e.args[0])
    print('Some other RsCmwGsmMeas error...')

finally:
    driver.utilities.visa_timeout = 5000
    # Close the session in any case
    driver.close()
```

Tip: General rules for exception handling:

- If you are sending commands that might generate errors in the instrument, for example deleting a file which does not exist, use the **OPTION 1** - temporarily disable status checking, send the command, clear the error queue and enable the status checking again.
- If you are sending queries that might generate errors or timeouts, for example querying measurement that can not be performed at the moment, use the **OPTION 2** - try/except with optionally adjusting the timeouts.

1.8 Transferring Files

Instrument -> PC

You definitely experienced it: you just did a perfect measurement, saved the results as a screenshot to an instrument's storage drive. Now you want to transfer it to your PC. With RsCmwGsmMeas, no problem, just figure out where the screenshot was stored on the instrument. In our case, it is *var/user/instr_screenshot.png*:

```
driver.utilities.read_file_from_instrument_to_pc(
    r'var/user/instr_screenshot.png',
    r'c:\temp\pc_screenshot.png')
```

PC -> Instrument

Another common scenario: Your cool test program contains a setup file you want to transfer to your instrument: Here is the RsCmwGsmMeas one-liner split into 3 lines:

```
driver.utilities.send_file_from_pc_to_instrument(
    r'c:\MyCoolTestProgram\instr_setup.sav',
    r'var/appdata/instr_setup.sav')
```

1.9 Writing Binary Data

Writing from bytes

An example where you need to send binary data is a waveform file of a vector signal generator. First, you compose your wform_data as `bytes`, and then you send it with `write_bin_block()`:

```
# MyWaveform.wv is an instrument file name under which this data is stored
driver.utilities.write_bin_block(
    "SOUR:BB:ARB:WAV:DATA 'MyWaveform.wv',",
    wform_data)
```

Note: Notice the `write_bin_block()` has two parameters:

- `string` parameter `cmd` for the SCPI command
- `bytes` parameter `payload` for the actual binary data to send

Writing from PC files

Similar to querying binary data to a file, you can write binary data from a file. The second parameter is then the PC file path the content of which you want to send:

```
driver.utilities.write_bin_block_from_file(
    "SOUR:BB:ARB:WAV:DATA 'MyWaveform.wv',",
    r"c:\temp\wform_data.wv")
```

1.10 Transferring Big Data with Progress

We can agree that it can be annoying using an application that shows no progress for long-lasting operations. The same is true for remote-control programs. Luckily, the RsCmwGsmMeas has this covered. And, this feature is quite universal - not just for big files transfer, but for any data in both directions.

RsCmwGsmMeas allows you to register a function (programmers fancy name is `callback`), which is then periodicaly invoked after transfer of one data chunk. You can define that chunk size, which gives you control over the callback invoke frequency. You can even slow down the transfer speed, if you want to process the data as they arrive (direction instrument -> PC).

To show this in praxis, we are going to use another *University-Professor-Example*: querying the ***IDN?** with chunk size of 2 bytes and delay of 200ms between each chunk read:

```
"""
Event handlers by reading
"""

from RsCmwGsmMeas import *
import time


def my_transfer_handler(args):
    """Function called each time a chunk of data is transferred"""
    # Total size is not always known at the beginning of the transfer
    total_size = args.total_size if args.total_size is not None else "unknown"

    print(f"Context: '{args.context}{'with opc' if args.opc_sync else ''}', "
        f"chunk {args.chunk_ix}, "
        f"transferred {args.transferred_size} bytes, "
        f"total size {total_size}, "
        f"direction {'reading' if args.reading else 'writing'}, "
        f"data '{args.data}'")

    if args.end_of_transfer:
        print('End of Transfer')
    time.sleep(0.2)


driver = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')

driver.events.on_read_handler = my_transfer_handler
# Switch on the data to be included in the event arguments
```

(continues on next page)

(continued from previous page)

```
# The event arguments args.data will be updated
driver.events.io_events_include_data = True
# Set data chunk size to 2 bytes
driver.utilities.data_chunk_size = 2
driver.utilities.query_str('*IDN?')
# Unregister the event handler
driver.utilities.on_read_handler = None

# Close the session
driver.close()
```

If you start it, you might wonder (or maybe not): why is the `args.total_size = None`? The reason is, in this particular case the RsCmwGsmMeas does not know the size of the complete response up-front. However, if you use the same mechanism for transfer of a known data size (for example, file transfer), you get the information about the total size too, and hence you can calculate the progress as:

*progress [pct] = 100 * args.transferred_size / args.total_size*

Snippet of transferring file from PC to instrument, the rest of the code is the same as in the previous example:

```
driver.events.on_write_handler = my_transfer_handler
driver.events.io_events_include_data = True
driver.data_chunk_size = 1000
driver.utilities.send_file_from_pc_to_instrument(
    r'c:\MyCoolTestProgram\my_big_file.bin',
    r'var/user/my_big_file.bin')
# Unregister the event handler
driver.events.on_write_handler = None
```

1.11 Multithreading

You are at the party, many people talking over each other. Not every person can deal with such crosstalk, neither can measurement instruments. For this reason, RsCmwGsmMeas has a feature of scheduling the access to your instrument by using so-called **Locks**. Locks make sure that there can be just one client at a time *talking* to your instrument. Talking in this context means completing one communication step - one command write or write/read or write/read/error check.

To describe how it works, and where it matters, we take three typical mulithread scenarios:

One instrument session, accessed from multiple threads

You are all set - the lock is a part of your instrument session. Check out the following example - it will execute properly, although the instrument gets 10 queries at the same time:

```
"""
Multiple threads are accessing one RsCmwGsmMeas object
"""

import threading
from RsCmwGsmMeas import *
```

(continues on next page)

(continued from previous page)

```
def execute(session):
    """Executed in a separate thread."""
    session.utilities.query_str('*IDN?')


driver = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')
threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver, ))
    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver.close()
```

Shared instrument session, accessed from multiple threads

Same as the previous case, you are all set. The session carries the lock with it. You have two objects, talking to the same instrument from multiple threads. Since the instrument session is shared, the same lock applies to both objects causing the exclusive access to the instrument.

Try the following example:

```
"""
Multiple threads are accessing two RsCmwGsmMeas objects with shared session
"""

import threading
from RsCmwGsmMeas import *


def execute(session: RsCmwGsmMeas, session_ix, index) -> None:
    """Executed in a separate thread."""
    print(f'{index} session {session_ix} query start...')
    session.utilities.query_str('*IDN?')
    print(f'{index} session {session_ix} query end')


driver1 = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCmwGsmMeas.from_existing_session(driver1)
driver1.utilities.visa_timeout = 200
driver2.utilities.visa_timeout = 200
# To see the effect of crosstalk, uncomment this line
# driver2.utilities.clear_lock()

threads = []
```

(continues on next page)

(continued from previous page)

```
for i in range(10):
    t = threading.Thread(target=execute, args=(driver1, 1, i,))
    t.start()
    threads.append(t)
    t = threading.Thread(target=execute, args=(driver2, 2, i,))
    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver2.close()
driver1.close()
```

As you see, everything works fine. If you want to simulate some party crosstalk, uncomment the line `driver2.utilities.clear_lock()`. Thich causes the driver2 session lock to break away from the driver1 session lock. Although the driver1 still tries to schedule its instrument access, the driver2 tries to do the same at the same time, which leads to all the fun stuff happening.

Multiple instrument sessions accessed from multiple threads

Here, there are two possible scenarios depending on the instrument's VISA interface:

- Your are lucky, because you instrument handles each remote session completely separately. An example of such instrument is SMW200A. In this case, you have no need for session locking.
- Your instrument handles all sessions with one set of in/out buffers. You need to lock the session for the duration of a talk. And you are lucky again, because the RsCmwGsmMeas takes care of it for you. The text below describes this scenario.

Run the following example:

```
"""
Multiple threads are accessing two RsCmwGsmMeas objects with two separate sessions
"""

import threading
from RsCmwGsmMeas import *


def execute(session: RsCmwGsmMeas, session_ix, index) -> None:
    """Executed in a separate thread."""
    print(f'{index} session {session_ix} query start...')
    session.utilities.query_str('*IDN?')
    print(f'{index} session {session_ix} query end')


driver1 = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCmwGsmMeas('TCPIP::192.168.56.101::INSTR')
driver1.utilities.visa_timeout = 200
```

(continues on next page)

(continued from previous page)

```
driver2.utilities.visa_timeout = 200

# Synchronise the sessions by sharing the same lock
driver2.utilities.assign_lock(driver1.utilities.get_lock())  # To see the effect of␣
↪crosstalk, comment this line

threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver1, 1, i,))
    t.start()
    threads.append(t)
    t = threading.Thread(target=execute, args=(driver2, 2, i,))
    t.start()
    threads.append(t)
print('All threads started')

# Wait for all threads to join this main thread
for t in threads:
    t.join()
print('All threads ended')

driver2.close()
driver1.close()
```

You have two completely independent sessions that want to talk to the same instrument at the same time. This will not go well, unless they share the same session lock. The key command to achieve this is `driver2.utilities.assign_lock(driver1.utilities.get_lock())` Try to comment it and see how it goes. If despite commenting the line the example runs without issues, you are lucky to have an instrument similar to the SMW200A.

CHAPTER

TWO

REVISION HISTORY

Rohde & Schwarz CMW Base System RsCmwBase instrument driver.

Supported instruments: CMW500, CMW100, CMW270, CMW280

The package is hosted here: https://pypi.org/project/RsCmwBase/

Documentation: https://RsCmwBase.readthedocs.io/

Examples: https://github.com/Rohde-Schwarz/Examples/

Currently supported CMW subsystems:

- Base: RsCmwBase
- Global Purpose RF: RsCmwGprfGen, RsCmwGprfMeas
- Bluetooth: RsCmwBluetoothSig, RsCmwBluetoothMeas
- LTE: RsCmwLteSig, RsCmwLteMeas
- CDMA2000: RsCdma2kSig, RsCdma2kMeas
- 1xEVDO: RsCmwEvdoSig, RsCmwEvdoMeas
- WCDMA: RsCmwWcdmaSig, RsCmwWcdmaMeas
- GSM: RsCmwGsmSig, RsCmwGsmMeas
- WLAN: RsCmwWlanSig, RscmwWlanMeas
- DAU: RsCMwDau

In case you require support for more subsystems, please contact our customer support on customersupport@rohde-schwarz.com with the topic "Auto-generated Python drivers" in the email subject. This will speed up the response process

---

Examples: Download the file 'CMW Python instrument drivers' from https://www.rohde-schwarz.com/driver/cmw500_overview/ The zip file contains the examples on how to use these drivers. Remember to adjust the resourceName string to fit your instrument.

Release Notes for the whole RsCmwXXX group:

Latest release notes summary: <INVALID>

Version 3.7.90.39

- <INVALID>

Version 3.8.xx2

- Fixed several misspelled arguments and command headers

Version 3.8.xx1

- Bluetooth and WLAN update for FW versions 3.8.xxx

Version 3.7.xx8

- Added documentation on ReadTheDocs

Version 3.7.xx7

- Added 3G measurement subsystems RsCmwGsmMeas, RsCmwCdma2kMeas, RsCmwEvdoMeas, RsCmwWcdmaMeas
- Added new data types for commands accepting numbers or ON/OFF:
- int or bool
- float or bool

Version 3.7.xx6

- Added new UDF integer number recognition

Version 3.7.xx5

- Added RsCmwDau

Version 3.7.xx4

- Fixed several interface names
- New release for CMW Base 3.7.90
- New release for CMW Bluetooth 3.7.90

Version 3.7.xx3

- Second release of the CMW python drivers packet
- New core component RsInstrument
- Previously, the groups starting with CATalog: e.g. 'CATalog:SIGNaling:TOPology:PLMN' were reordered to 'SIGNaling:TOPology:PLMN:CATALOG' give more contextual meaning to the method/property name. This is now reverted back, since it was hard to find the desired functionality.
- Reorganized Utilities interface to sub-groups

Version 3.7.xx2

- Fixed some misspeling errors
- Changed enum and repCap types names
- All the assemblies are signed with Rohde & Schwarz signature

Version 1.0.0.0

- First released version

CHAPTER

THREE

ENUMS

3.1 AcquisitionMode

```
# Example value:
value = enums.AcquisitionMode.GAP
# All values (2x):
GAP | PATTern
```

3.2 Band

```
# Example value:
value = enums.Band.G04
# All values (6x):
G04 | G085 | G09 | G18 | G19 | GG08
```

3.3 CmwsConnector

```
# First value:
value = enums.CmwsConnector.R11
# Last value:
value = enums.CmwsConnector.RB8
# All values (48x):
R11 | R12 | R13 | R14 | R15 | R16 | R17 | R18
R21 | R22 | R23 | R24 | R25 | R26 | R27 | R28
R31 | R32 | R33 | R34 | R35 | R36 | R37 | R38
R41 | R42 | R43 | R44 | R45 | R46 | R47 | R48
RA1 | RA2 | RA3 | RA4 | RA5 | RA6 | RA7 | RA8
RB1 | RB2 | RB3 | RB4 | RB5 | RB6 | RB7 | RB8
```

3.4 Decode

```
# Example value:
value = enums.Decode.GTBits
# All values (2x):
GTBits | STANdard
```

3.5 FilterIq

```
# Example value:
value = enums.FilterIq.F90Khz
# All values (3x):
F90Khz | ISIRemoved | UNFiltered
```

3.6 FilterPvTime

```
# Example value:
value = enums.FilterPvTime.G05M
# All values (2x):
G05M | G10M
```

3.7 ListMode

```
# Example value:
value = enums.ListMode.ONCE
# All values (2x):
ONCE | SEGMent
```

3.8 LoopType

```
# Example value:
value = enums.LoopType.C
# All values (2x):
C | SRB
```

3.9 ParameterSetMode

```
# Example value:
value = enums.ParameterSetMode.GLOBal
# All values (2x):
GLOBal | LIST
```

3.10 PclMode

```
# Example value:
value = enums.PclMode.AUTO
# All values (3x):
AUTO | PCL | SIGNaling
```

3.11 PeakHoldMode

```
# Example value:
value = enums.PeakHoldMode.PHOL
# All values (2x):
PHOL | SCO
```

3.12 RangeMode

```
# Example value:
value = enums.RangeMode.NORMal
# All values (2x):
NORMal | WIDE
```

3.13 RefPowerMode

```
# Example value:
value = enums.RefPowerMode.AVERage
# All values (3x):
AVERage | CURRent | DCOMpensated
```

3.14 Repeat

```
# Example value:
value = enums.Repeat.CONTinuous
# All values (2x):
CONTinuous | SINGleshot
```

3.15 ResourceState

```
# Example value:
value = enums.ResourceState.ACTive
# All values (8x):
ACTive | ADJusted | INValid | OFF | PENDing | QUEued | RDY | RUN
```

3.16 ResultStatus2

```
# First value:
value = enums.ResultStatus2.DC
# Last value:
value = enums.ResultStatus2.ULEU
# All values (10x):
DC | INV | NAV | NCAP | OFF | OFL | OK | UFL
ULEL | ULEU
```

3.17 RetriggerFlag

```
# Example value:
value = enums.RetriggerFlag.IFPower
# All values (3x):
IFPower | OFF | ON
```

3.18 RfConverter

```
# First value:
value = enums.RfConverter.IRX1
# Last value:
value = enums.RfConverter.RX44
# All values (40x):
IRX1 | IRX11 | IRX12 | IRX13 | IRX14 | IRX2 | IRX21 | IRX22
IRX23 | IRX24 | IRX3 | IRX31 | IRX32 | IRX33 | IRX34 | IRX4
IRX41 | IRX42 | IRX43 | IRX44 | RX1 | RX11 | RX12 | RX13
RX14 | RX2 | RX21 | RX22 | RX23 | RX24 | RX3 | RX31
RX32 | RX33 | RX34 | RX4 | RX41 | RX42 | RX43 | RX44
```

3.19 Rotation

```
# Example value:
value = enums.Rotation.P38
# All values (2x):
P38 | P38R
```

3.20 RxConnector

```
# First value:
value = enums.RxConnector.I11I
# Last value:
value = enums.RxConnector.RH8
# All values (154x):
I11I | I13I | I15I | I17I | I21I | I23I | I25I | I27I
I31I | I33I | I35I | I37I | I41I | I43I | I45I | I47I
IF1 | IF2 | IF3 | IQ1I | IQ3I | IQ5I | IQ7I | R11
R11C | R12 | R12C | R12I | R13 | R13C | R14 | R14C
R14I | R15 | R16 | R17 | R18 | R21 | R21C | R22
R22C | R22I | R23 | R23C | R24 | R24C | R24I | R25
R26 | R27 | R28 | R31 | R31C | R32 | R32C | R32I
R33 | R33C | R34 | R34C | R34I | R35 | R36 | R37
R38 | R41 | R41C | R42 | R42C | R42I | R43 | R43C
R44 | R44C | R44I | R45 | R46 | R47 | R48 | RA1
RA2 | RA3 | RA4 | RA5 | RA6 | RA7 | RA8 | RB1
RB2 | RB3 | RB4 | RB5 | RB6 | RB7 | RB8 | RC1
RC2 | RC3 | RC4 | RC5 | RC6 | RC7 | RC8 | RD1
RD2 | RD3 | RD4 | RD5 | RD6 | RD7 | RD8 | RE1
RE2 | RE3 | RE4 | RE5 | RE6 | RE7 | RE8 | RF1
RF1C | RF2 | RF2C | RF2I | RF3 | RF3C | RF4 | RF4C
RF4I | RF5 | RF5C | RF6 | RF6C | RF7 | RF8 | RFAC
RFBC | RFBI | RG1 | RG2 | RG3 | RG4 | RG5 | RG6
RG7 | RG8 | RH1 | RH2 | RH3 | RH4 | RH5 | RH6
RH7 | RH8
```

3.21 Scenario

```
# Example value:
value = enums.Scenario.CSPath
# All values (4x):
CSPath | MAPRotocol | NAV | SALone
```

3.22 SignalSlope

```
# Example value:
value = enums.SignalSlope.FEDGe
# All values (2x):
FEDGe | REDGe
```

3.23 SlotA

```
# Example value:
value = enums.SlotA.ACCess
# All values (6x):
ACCess | ANY | EPSK | GMSK | OFF | Q16
```

3.24 SlotB

```
# Example value:
value = enums.SlotB.EPSK
# All values (3x):
EPSK | GMSK | OFF
```

3.25 SlotInfo

```
# Example value:
value = enums.SlotInfo.ACCess
# All values (5x):
ACCess | EPSK | GMSK | OFF | Q16
```

3.26 StopCondition

```
# Example value:
value = enums.StopCondition.NONE
# All values (2x):
NONE | SLFail
```

3.27 TscA

```
# First value:
value = enums.TscA.DUMM
# Last value:
value = enums.TscA.TSCA
# All values (11x):
DUMM | OFF | TSC0 | TSC1 | TSC2 | TSC3 | TSC4 | TSC5
TSC6 | TSC7 | TSCA
```

3.28 TscB

```
# First value:
value = enums.TscB.AB0
# Last value:
value = enums.TscB.OFF
# All values (18x):
AB0 | AB1 | AB2 | AB3 | AB4 | AB5 | AB6 | AB7
DUMMy | NB0 | NB1 | NB2 | NB3 | NB4 | NB5 | NB6
NB7 | OFF
```

CHAPTER

FOUR

REPCAPS

4.1 Instance (Global)

```
# Setting:
driver.repcap_instance_set(repcap.Instance.Inst1)
# Range:
Inst1 .. Inst16
# All values (16x):
Inst1 | Inst2 | Inst3 | Inst4 | Inst5 | Inst6 | Inst7 | Inst8
Inst9 | Inst10 | Inst11 | Inst12 | Inst13 | Inst14 | Inst15 | Inst16
```

4.2 AbPower

```
# First value:
value = repcap.AbPower.Nr1
# Range:
Nr1 .. Nr10
# All values (10x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10
```

4.3 FallingEdge

```
# First value:
value = repcap.FallingEdge.Nr1
# Values (4x):
Nr1 | Nr2 | Nr3 | Nr4
```

4.4 FreqOffset

```
# First value:
value = repcap.FreqOffset.Nr1
# Range:
Nr1 .. Nr41
# All values (41x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40
Nr41
```

4.5 MeasPoint

```
# First value:
value = repcap.MeasPoint.Nr1
# Range:
Nr1 .. Nr20
# All values (20x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20
```

4.6 QamOrder

```
# First value:
value = repcap.QamOrder.Nr16
# Values (1x):
Nr16
```

4.7 RangePcl

```
# First value:
value = repcap.RangePcl.Nr1
# Range:
Nr1 .. Nr5
# All values (5x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5
```

4.8 RisingEdge

```
# First value:
value = repcap.RisingEdge.Nr1
# Values (4x):
Nr1 | Nr2 | Nr3 | Nr4
```

4.9 Segment

```
# First value:
value = repcap.Segment.Nr1
# Range:
Nr1 .. Nr512
# All values (512x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20 | Nr21 | Nr22 | Nr23 | Nr24
Nr25 | Nr26 | Nr27 | Nr28 | Nr29 | Nr30 | Nr31 | Nr32
Nr33 | Nr34 | Nr35 | Nr36 | Nr37 | Nr38 | Nr39 | Nr40
Nr41 | Nr42 | Nr43 | Nr44 | Nr45 | Nr46 | Nr47 | Nr48
Nr49 | Nr50 | Nr51 | Nr52 | Nr53 | Nr54 | Nr55 | Nr56
Nr57 | Nr58 | Nr59 | Nr60 | Nr61 | Nr62 | Nr63 | Nr64
Nr65 | Nr66 | Nr67 | Nr68 | Nr69 | Nr70 | Nr71 | Nr72
Nr73 | Nr74 | Nr75 | Nr76 | Nr77 | Nr78 | Nr79 | Nr80
Nr81 | Nr82 | Nr83 | Nr84 | Nr85 | Nr86 | Nr87 | Nr88
Nr89 | Nr90 | Nr91 | Nr92 | Nr93 | Nr94 | Nr95 | Nr96
Nr97 | Nr98 | Nr99 | Nr100 | Nr101 | Nr102 | Nr103 | Nr104
Nr105 | Nr106 | Nr107 | Nr108 | Nr109 | Nr110 | Nr111 | Nr112
Nr113 | Nr114 | Nr115 | Nr116 | Nr117 | Nr118 | Nr119 | Nr120
Nr121 | Nr122 | Nr123 | Nr124 | Nr125 | Nr126 | Nr127 | Nr128
Nr129 | Nr130 | Nr131 | Nr132 | Nr133 | Nr134 | Nr135 | Nr136
Nr137 | Nr138 | Nr139 | Nr140 | Nr141 | Nr142 | Nr143 | Nr144
Nr145 | Nr146 | Nr147 | Nr148 | Nr149 | Nr150 | Nr151 | Nr152
Nr153 | Nr154 | Nr155 | Nr156 | Nr157 | Nr158 | Nr159 | Nr160
Nr161 | Nr162 | Nr163 | Nr164 | Nr165 | Nr166 | Nr167 | Nr168
Nr169 | Nr170 | Nr171 | Nr172 | Nr173 | Nr174 | Nr175 | Nr176
Nr177 | Nr178 | Nr179 | Nr180 | Nr181 | Nr182 | Nr183 | Nr184
Nr185 | Nr186 | Nr187 | Nr188 | Nr189 | Nr190 | Nr191 | Nr192
Nr193 | Nr194 | Nr195 | Nr196 | Nr197 | Nr198 | Nr199 | Nr200
Nr201 | Nr202 | Nr203 | Nr204 | Nr205 | Nr206 | Nr207 | Nr208
Nr209 | Nr210 | Nr211 | Nr212 | Nr213 | Nr214 | Nr215 | Nr216
Nr217 | Nr218 | Nr219 | Nr220 | Nr221 | Nr222 | Nr223 | Nr224
Nr225 | Nr226 | Nr227 | Nr228 | Nr229 | Nr230 | Nr231 | Nr232
Nr233 | Nr234 | Nr235 | Nr236 | Nr237 | Nr238 | Nr239 | Nr240
Nr241 | Nr242 | Nr243 | Nr244 | Nr245 | Nr246 | Nr247 | Nr248
Nr249 | Nr250 | Nr251 | Nr252 | Nr253 | Nr254 | Nr255 | Nr256
Nr257 | Nr258 | Nr259 | Nr260 | Nr261 | Nr262 | Nr263 | Nr264
Nr265 | Nr266 | Nr267 | Nr268 | Nr269 | Nr270 | Nr271 | Nr272
Nr273 | Nr274 | Nr275 | Nr276 | Nr277 | Nr278 | Nr279 | Nr280
Nr281 | Nr282 | Nr283 | Nr284 | Nr285 | Nr286 | Nr287 | Nr288
```

(continues on next page)

(continued from previous page)

```
Nr289 | Nr290 | Nr291 | Nr292 | Nr293 | Nr294 | Nr295 | Nr296
Nr297 | Nr298 | Nr299 | Nr300 | Nr301 | Nr302 | Nr303 | Nr304
Nr305 | Nr306 | Nr307 | Nr308 | Nr309 | Nr310 | Nr311 | Nr312
Nr313 | Nr314 | Nr315 | Nr316 | Nr317 | Nr318 | Nr319 | Nr320
Nr321 | Nr322 | Nr323 | Nr324 | Nr325 | Nr326 | Nr327 | Nr328
Nr329 | Nr330 | Nr331 | Nr332 | Nr333 | Nr334 | Nr335 | Nr336
Nr337 | Nr338 | Nr339 | Nr340 | Nr341 | Nr342 | Nr343 | Nr344
Nr345 | Nr346 | Nr347 | Nr348 | Nr349 | Nr350 | Nr351 | Nr352
Nr353 | Nr354 | Nr355 | Nr356 | Nr357 | Nr358 | Nr359 | Nr360
Nr361 | Nr362 | Nr363 | Nr364 | Nr365 | Nr366 | Nr367 | Nr368
Nr369 | Nr370 | Nr371 | Nr372 | Nr373 | Nr374 | Nr375 | Nr376
Nr377 | Nr378 | Nr379 | Nr380 | Nr381 | Nr382 | Nr383 | Nr384
Nr385 | Nr386 | Nr387 | Nr388 | Nr389 | Nr390 | Nr391 | Nr392
Nr393 | Nr394 | Nr395 | Nr396 | Nr397 | Nr398 | Nr399 | Nr400
Nr401 | Nr402 | Nr403 | Nr404 | Nr405 | Nr406 | Nr407 | Nr408
Nr409 | Nr410 | Nr411 | Nr412 | Nr413 | Nr414 | Nr415 | Nr416
Nr417 | Nr418 | Nr419 | Nr420 | Nr421 | Nr422 | Nr423 | Nr424
Nr425 | Nr426 | Nr427 | Nr428 | Nr429 | Nr430 | Nr431 | Nr432
Nr433 | Nr434 | Nr435 | Nr436 | Nr437 | Nr438 | Nr439 | Nr440
Nr441 | Nr442 | Nr443 | Nr444 | Nr445 | Nr446 | Nr447 | Nr448
Nr449 | Nr450 | Nr451 | Nr452 | Nr453 | Nr454 | Nr455 | Nr456
Nr457 | Nr458 | Nr459 | Nr460 | Nr461 | Nr462 | Nr463 | Nr464
Nr465 | Nr466 | Nr467 | Nr468 | Nr469 | Nr470 | Nr471 | Nr472
Nr473 | Nr474 | Nr475 | Nr476 | Nr477 | Nr478 | Nr479 | Nr480
Nr481 | Nr482 | Nr483 | Nr484 | Nr485 | Nr486 | Nr487 | Nr488
Nr489 | Nr490 | Nr491 | Nr492 | Nr493 | Nr494 | Nr495 | Nr496
Nr497 | Nr498 | Nr499 | Nr500 | Nr501 | Nr502 | Nr503 | Nr504
Nr505 | Nr506 | Nr507 | Nr508 | Nr509 | Nr510 | Nr511 | Nr512
```

4.10 SubVector

```
# First value:
value = repcap.SubVector.Nr1
# Range:
Nr1 .. Nr12
# All values (12x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12
```

4.11 UsefulPart

```
# First value:
value = repcap.UsefulPart.Nr1
# Range:
Nr1 .. Nr5
# All values (5x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5
```

CHAPTER

FIVE

EXAMPLES

For more examples, visit our Rohde & Schwarz Github repository.

```
""" Example on how to use the python RsCmw auto-generated instrument driver showing:
- usage of basic properties of the cmw_base object
- basic concept of setting commands and repcaps: DISPlay:WINDow<n>:SELect
- cmw_xxx drivers reliability interface usage
"""

from RsCmwBase import *  # install from pypi.org

RsCmwBase.assert_minimum_version('3.7.90.32')
cmw_base = RsCmwBase('TCPIP::10.112.1.116::INSTR', True, False)
print(f'CMW Base IND: {cmw_base.utilities.idn_string}')
print(f'CMW Instrument options:\n{",".join(cmw_base.utilities.instrument_options)}')
cmw_base.utilities.visa_timeout = 5000

# Sends OPC after each command
cmw_base.utilities.opc_query_after_write = False

# Checks for syst:err? after each command / query
cmw_base.utilities.instrument_status_checking = True

# DISPlay:WINDow<n>:SELect
cmw_base.display.window.select.set(repcap.Window.Win1)
cmw_base.display.window.repcap_window_set(repcap.Window.Win2)
cmw_base.display.window.select.set()

# Self-test
self_test = cmw_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmw_base.reliability.ExceptionOnError = True


# Callback to use for the reliability indicator update event
def my_reliability_handler(event_args: ReliabilityEventArgs):
        print(f'Base Reliability updated.\nContext: {event_args.context}\nMessage:
↪{event_args.message}')
```

(continues on next page)

(continued from previous page)

```
# We register a callback for each change in the reliability indicator
cmw_base.reliability.on_update_handler = my_reliability_handler

# You can obtain the last value of the returned reliability
print(f"\nReliability last value: {cmw_base.reliability.last_value}, context '{cmw_base.
↪reliability.last_context}', message: {cmw_base.reliability.last_message}")

# Reference Frequency Source
cmw_base.system.reference.frequency.source_set(enums.SourceIntExt.INTernal)

# Close the session
cmw_base.close()
```

CHAPTER

SIX

INDEX

CHAPTER

SEVEN

RSCMWGSMMEAS API STRUCTURE

Global RepCaps

```
driver = RsCmwGsmMeas('TCPIP::192.168.2.101::HISLIP')
# Instance range: Inst1 .. Inst16
rc = driver.repcap_instance_get()
driver.repcap_instance_set(repcap.Instance.Inst1)
```

class RsCmwGsmMeas(*resource_name: str*, *id_query: bool = True*, *reset: bool = False*, *options: Optional[str] = None*, *direct_session: Optional[object] = None*)

378 total commands, 4 Sub-groups, 0 group commands

Initializes new RsCmwGsmMeas session.

Parameter options tokens examples:

- 'Simulate=True' - starts the session in simulation mode. Default: False
- 'SelectVisa=socket' - uses no VISA implementation for socket connections - you do not need any VISA-C installation
- 'SelectVisa=rs' - forces usage of RohdeSchwarz Visa
- 'SelectVisa=ni' - forces usage of National Instruments Visa
- 'QueryInstrumentStatus = False' - same as driver.utilities.instrument_status_checking = False
- 'DriverSetup=(WriteDelay = 20, ReadDelay = 5)' - Introduces delay of 20ms before each write and 5ms before each read
- 'DriverSetup=(OpcWaitMode = OpcQuery)' - mode for all the opc-synchronised write/reads. Other modes: StbPolling, StbPollingSlow, StbPollingSuperSlow
- 'DriverSetup=(AddTermCharToWriteBinBLock = True)' - Adds one additional LF to the end of the binary data (some instruments require that)
- 'DriverSetup=(AssureWriteWithTermChar = True)' - Makes sure each command/query is terminated with termination character. Default: Interface dependent
- 'DriverSetup=(TerminationCharacter = 'x')' - Sets the termination character for reading. Default: '<LF>' (LineFeed)
- 'DriverSetup=(IoSegmentSize = 10E3)' - Maximum size of one write/read segment. If transferred data is bigger, it is split to more segments
- 'DriverSetup=(OpcTimeout = 10000)' - same as driver.utilities.opc_timeout = 10000
- 'DriverSetup=(VisaTimeout = 5000)' - same as driver.utilities.visa_timeout = 5000

- 'DriverSetup=(ViClearExeMode = 255)' - Binary combination where 1 means performing viClear() on a certain interface as the very first command in init
- 'DriverSetup=(OpcQueryAfterWrite = True)' - same as driver.utilities.opc_query_after_write = True

Parameters

- **resource_name** – VISA resource name, e.g. 'TCPIP::192.168.2.1::INSTR'
- **id_query** – if True: the instrument's model name is verified against the models supported by the driver and eventually throws an exception.
- **reset** – Resets the instrument (sends *RST command) and clears its status sybsystem
- **options** – string tokens alternating the driver settings.
- **direct_session** – Another driver object or pyVisa object to reuse the session instead of opening a new session.

static assert_minimum_version(*min_version: str*) → None

Asserts that the driver version fulfills the minimum required version you have entered. This way you make sure your installed driver is of the entered version or newer.

close() → None

Closes the active RsCmwGsmMeas session.

classmethod from_existing_session(*session: object*, *options: Optional[str] = None*) → RsCmwGsmMeas

Creates a new RsCmwGsmMeas object with the entered 'session' reused.

Parameters

- **session** – can be an another driver or a direct pyvisa session.
- **options** – string tokens alternating the driver settings.

get_session_handle() → object

Returns the underlying session handle.

static list_resources(*expression: str = '?*::INSTR'*, *visa_select: Optional[str] = None*) → List[str]

Finds all the resources defined by the expression

- '?*' - matches all the available instruments
- 'USB::?*' - matches all the USB instruments
- "TCPIP::192?*' - matches all the LAN instruments with the IP address starting with 192

Parameters

- **expression** – see the examples in the function
- **visa_select** – optional parameter selecting a specific VISA. Examples: '@ni', '@rs'

restore_all_repcaps_to_default() → None

Sets all the Group and Global repcaps to their initial values

Subgroups

7.1 Route

SCPI Commands

```
ROUTe:GSM:MEASurement<Instance>
```

class Route

Route commands group definition. 5 total commands, 1 Sub-groups, 1 group commands

class ValueStruct

Structure for reading output parameters. Fields:

- Scenario: enums.Scenario: SALone | CSPath | MAPRotocol SALone: Standalone (non-signaling) CSPath: Combined signal path MAPRotocol: Measure@Protocol test
- Controller: str: string Controlling application for scenario CSPath or MAPRotocol
- Rx_Connector: enums.RxConnector: RF connector for the input path
- Rf_Converter: enums.RfConverter: RX module for the input path

get_value() → ValueStruct

```
# SCPI: ROUTe:GSM:MEASurement<Instance>
value: ValueStruct = driver.route.get_value()
```

Returns the configured routing settings. For possible connector and converter values, see 'Values for RF Path Selection'.

return structure: for return value, see the help for ValueStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.clone()
```

Subgroups

7.1.1 Scenario

SCPI Commands

```
ROUTe:GSM:MEASurement<Instance>:SCENario:CSPath
ROUTe:GSM:MEASurement<Instance>:SCENario:SALone
ROUTe:GSM:MEASurement<Instance>:SCENario
```

class Scenario

Scenario commands group definition. 4 total commands, 1 Sub-groups, 3 group commands

class SaloneStruct

Structure for reading output parameters. Fields:

- Rx_Connector: enums.RxConnector: RF connector for the input path
- Rf_Converter: enums.RfConverter: RX module for the input path

get_cspath() → str

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SCENario:CSPath
value: str = driver.route.scenario.get_cspath()
```

Activates the combined signal path scenario and selects a master firmware application for the GSM measurements. The master controls the signal routing settings and analyzer settings while the combined signal path scenario is active.

return master: string String parameter containing the master application, e.g. 'GSM Sig1' or 'GSM Sig2'

get_salone() → SaloneStruct

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SCENario:SALone
value: SaloneStruct = driver.route.scenario.get_salone()
```

Activates the standalone scenario and selects the RF input path for the measured RF signal. For possible connector and converter values, see 'Values for RF Path Selection'.

return structure: for return value, see the help for SaloneStruct structure arguments.

get_value() → RsCmwGsmMeas.enums.Scenario

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SCENario
value: enums.Scenario = driver.route.scenario.get_value()
```

Queries the active scenario.

return scenario: SALone | CSPath | MAPRotocol Standalone, combined signal path, measure at protocol test

set_cspath(*master: str*) → None

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SCENario:CSPath
driver.route.scenario.set_cspath(master = '1')
```

Activates the combined signal path scenario and selects a master firmware application for the GSM measurements. The master controls the signal routing settings and analyzer settings while the combined signal path scenario is active.

param master string String parameter containing the master application, e.g. 'GSM Sig1' or 'GSM Sig2'

set_salone(*value: RsCmwGsmMeas.Implementations.Route_.Scenario.Scenario.SaloneStruct*) → None

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SCENario:SALone
driver.route.scenario.set_salone(value = SaloneStruct())
```

Activates the standalone scenario and selects the RF input path for the measured RF signal. For possible connector and converter values, see 'Values for RF Path Selection'.

param value see the help for SaloneStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.scenario.clone()
```

Subgroups

7.1.1.1 MaProtocol

SCPI Commands

```
ROUTe:GSM:MEASurement<Instance>:SCENario:MAPRotocol
```

class MaProtocol

MaProtocol commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set(*controler: Optional[str] = None*) → None

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SCENario:MAPRotocol
driver.route.scenario.maProtocol.set(controler = '1')
```

Activates the Measure@ProtocolTest scenario and optionally selects the controlling protocol test application. The signal routing and analyzer settings are ignored by the measurement application. Configure the corresponding settings within the protocol test application used in parallel.

param controler string String parameter selecting the protocol test application e.g., 'Protocol Test1'

7.2 Configure

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:BAND
CONFigure:GSM:MEASurement<Instance>:CHANnel
```

class Configure

Configure commands group definition. 130 total commands, 2 Sub-groups, 2 group commands

get_band() → RsCmwGsmMeas.enums.Band

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:BAND
value: enums.Band = driver.configure.get_band()
```

Selects the GSM frequency band. INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:BAND:BCCH
- SENSe:GSM:SIGN<i>:BAND:TCH

return band: G04 | G085 | G09 | G18 | G19 | GG08 G04: GSM400 G085: GSM850 G09: GSM900 G18: GSM1800 G19: GSM1900 GG08: GSMGT800

get_channel() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:CHANnel
value: int = driver.configure.get_channel()
```

Selects the channel number. The channel number must be valid for the current frequency band, for dependencies see 'GSM Frequency Bands and Channels'. The corresponding center frequency (method RsCmwGsmMeas.Configure.RfSettings.frequency) is calculated and set.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:CHANnel
- CONFigure:GSM:SIGN<i>:RFSettings:CHCCombined:TCH:CSWitched
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:ENABle
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:MAIO
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:HSN
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:SEQuence

return channel: decimal GSM channel number Range: depends on frequency band

set_band(*band: RsCmwGsmMeas.enums.Band*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:BAND
driver.configure.set_band(band = enums.Band.G04)
```

Selects the GSM frequency band. INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:BAND:BCCH
- SENSe:GSM:SIGN<i>:BAND:TCH

param band G04 | G085 | G09 | G18 | G19 | GG08 G04: GSM400 G085: GSM850 G09: GSM900 G18: GSM1800 G19: GSM1900 GG08: GSMGT800

set_channel(*channel: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:CHANnel
driver.configure.set_channel(channel = 1)
```

Selects the channel number. The channel number must be valid for the current frequency band, for dependencies see 'GSM Frequency Bands and Channels'. The corresponding center frequency (method RsCmwGsmMeas.Configure.RfSettings.frequency) is calculated and set.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:CHANnel
- CONFigure:GSM:SIGN<i>:RFSettings:CHCCombined:TCH:CSWitched
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:ENABle
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:MAIO

- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:HSN
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:SEQuence

param channel decimal GSM channel number Range: depends on frequency band

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.clone()
```

Subgroups

7.2.1 RfSettings

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:RFSettings:EATTenuation
CONFigure:GSM:MEASurement<Instance>:RFSettings:UMARgin
CONFigure:GSM:MEASurement<Instance>:RFSettings:ENPower
CONFigure:GSM:MEASurement<Instance>:RFSettings:FREQuency
CONFigure:GSM:MEASurement<Instance>:RFSettings:FOFFset
CONFigure:GSM:MEASurement<Instance>:RFSettings:MLOFfset
```

class RfSettings

RfSettings commands group definition. 6 total commands, 0 Sub-groups, 6 group commands

get_eattenuation() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:EATTenuation
value: float = driver.configure.rfSettings.get_eattenuation()
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:RFSettings:EATTenuation:INPut.

return external_att: numeric Range: -50 dB to 90 dB, Unit: dB

get_envelope_power() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:ENPower
value: float = driver.configure.rfSettings.get_envelope_power()
```

Sets the expected nominal power of the measured RF signal. INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:ENPMode
- CONFigure:GSM:SIGN<i>:RFSettings:ENPower

return exp_nom_power: numeric The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin The input power range is stated in the data sheet. Unit: dBm

get_foffset() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:FOFFset
value: int = driver.configure.rfSettings.get_foffset()
```

Specifies a positive or negative frequency offset to be added to the center frequency of the configured channel.

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:FOFFset:UL
- CONFigure:GSM:SIGN<i>:CONNection:RFOFfset

return offset: numeric Range: -100000 Hz to 100000 Hz , Unit: Hz

get_frequency() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:FREQuency
value: float = driver.configure.rfSettings.get_frequency()
```

Selects the center frequency of the RF analyzer. If the center frequency is valid for the current frequency band, the corresponding channel number is also calculated and set.

INTRO_CMD_HELP: See also:

- 'GSM Frequency Bands and Channels'
- method RsCmwGsmMeas.Configure.band
- method RsCmwGsmMeas.Configure.channel

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:CHANnel
- CONFigure:GSM:SIGN<i>:RFSettings:CHCCombined:TCH:CSWitched
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:ENABle
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:MAIO
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:HSN
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:SEQuence

The supported frequency range depends on the instrument model and the available options. The supported range can be smaller than stated here. Refer to the preface of your model-specific base unit manual.

return frequency: numeric Range: 70 MHz to 6 GHz , Unit: Hz

get_ml_offset() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:MLOFfset
value: float = driver.configure.rfSettings.get_ml_offset()
```

Varies the input level of the mixer in the analyzer path. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:RFSettings:MLOFfset.

return mix_lev_offset: numeric Range: -10 dB to 10 dB, Unit: dB

get_umargin() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:UMARgin
value: float = driver.configure.rfSettings.get_umargin()
```

Sets the margin that the measurement adds to the expected nominal power to determine the reference power. The reference power minus the external input attenuation must be within the power range of the selected input connector. Refer to the data sheet. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:RFSettings:UMARgin.

return user_margin: numeric Range: 0 dB to (55 dB + External Attenuation - Expected Nominal Power) , Unit: dB

set_eattenuation(*external_att: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:EATTenuation
driver.configure.rfSettings.set_eattenuation(external_att = 1.0)
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:RFSettings:EATTenuation:INPut.

param external_att numeric Range: -50 dB to 90 dB, Unit: dB

set_envelope_power(*exp_nom_power: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:ENPower
driver.configure.rfSettings.set_envelope_power(exp_nom_power = 1.0)
```

Sets the expected nominal power of the measured RF signal. INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:ENPMode
- CONFigure:GSM:SIGN<i>:RFSettings:ENPower

param exp_nom_power numeric The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin The input power range is stated in the data sheet. Unit: dBm

set_foffset(*offset: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:FOFFset
driver.configure.rfSettings.set_foffset(offset = 1)
```

Specifies a positive or negative frequency offset to be added to the center frequency of the configured channel. INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:FOFFset:UL
- CONFigure:GSM:SIGN<i>:CONNection:RFOFfset

param offset numeric Range: -100000 Hz to 100000 Hz , Unit: Hz

set_frequency(*frequency: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:FREQuency
driver.configure.rfSettings.set_frequency(frequency = 1.0)
```

Selects the center frequency of the RF analyzer. If the center frequency is valid for the current frequency band, the corresponding channel number is also calculated and set.

INTRO_CMD_HELP: See also:

- 'GSM Frequency Bands and Channels'
- method RsCmwGsmMeas.Configure.band
- method RsCmwGsmMeas.Configure.channel

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:CHANnel
- CONFigure:GSM:SIGN<i>:RFSettings:CHCCombined:TCH:CSWitched
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:ENABle
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:MAIO
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:HSN
- CONFigure:GSM:SIGN<i>:RFSettings:HOPPing:SEQuence

The supported frequency range depends on the instrument model and the available options. The supported range can be smaller than stated here. Refer to the preface of your model-specific base unit manual.

param frequency numeric Range: 70 MHz to 6 GHz , Unit: Hz

set_ml_offset(*mix_lev_offset: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:MLOFfset
driver.configure.rfSettings.set_ml_offset(mix_lev_offset = 1.0)
```

Varies the input level of the mixer in the analyzer path. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:RFSettings:MLOFfset.

param mix_lev_offset numeric Range: -10 dB to 10 dB, Unit: dB

set_umargin(*user_margin: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:RFSettings:UMARgin
driver.configure.rfSettings.set_umargin(user_margin = 1.0)
```

Sets the margin that the measurement adds to the expected nominal power to determine the reference power. The reference power minus the external input attenuation must be within the power range of the selected input connector. Refer to the data sheet. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:RFSettings:UMARgin.

param user_margin numeric Range: 0 dB to (55 dB + External Attenuation - Expected Nominal Power) , Unit: dB

7.2.2 MultiEval

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:TOUT
CONFigure:GSM:MEASurement<Instance>:MEValuation:REPetition
CONFigure:GSM:MEASurement<Instance>:MEValuation:SCONdition
CONFigure:GSM:MEASurement<Instance>:MEValuation:MOEXception
CONFigure:GSM:MEASurement<Instance>:MEValuation:RPMode
CONFigure:GSM:MEASurement<Instance>:MEValuation:FCRange
CONFigure:GSM:MEASurement<Instance>:MEValuation:HDALevel
CONFigure:GSM:MEASurement<Instance>:MEValuation:MSLots
CONFigure:GSM:MEASurement<Instance>:MEValuation:TSEQuence
CONFigure:GSM:MEASurement<Instance>:MEValuation:NBQSearch
CONFigure:GSM:MEASurement<Instance>:MEValuation:ABSearch
CONFigure:GSM:MEASurement<Instance>:MEValuation:MVIew
CONFigure:GSM:MEASurement<Instance>:MEValuation:AMODe
CONFigure:GSM:MEASurement<Instance>:MEValuation:APATtern
CONFigure:GSM:MEASurement<Instance>:MEValuation:GLENgth
CONFigure:GSM:MEASurement<Instance>:MEValuation:PCLMode
CONFigure:GSM:MEASurement<Instance>:MEValuation:PCL
CONFigure:GSM:MEASurement<Instance>:MEValuation:IIOFrames
```

class MultiEval

MultiEval commands group definition. 122 total commands, 11 Sub-groups, 18 group commands

class ApatternStruct

Structure for reading output parameters. Fields:

- Slot_0: enums.SlotB: No parameter help available
- Slot_1: enums.SlotB: No parameter help available
- Slot_2: enums.SlotB: No parameter help available
- Slot_3: enums.SlotB: No parameter help available
- Slot_4: enums.SlotB: No parameter help available
- Slot_5: enums.SlotB: No parameter help available
- Slot_6: enums.SlotB: No parameter help available
- Slot_7: enums.SlotB: No parameter help available

class MslotsStruct

Structure for reading output parameters. Fields:

- Slot_Offset: int: decimal Start of the measurement interval relative to the GSM frame boundary Range: 0 to 7
- Slot_Count: int: decimal Number of slots to be measured Range: 1 to 8
- Meas_Slot: int: decimal Slot to be measured for one-slot measurements Range: 0 to 7

class PclStruct

Structure for reading output parameters. Fields:

- Slot_0: int: integer Range: 0 to 31
- Slot_1: int: integer Range: 0 to 31

- Slot_2: int: integer Range: 0 to 31
- Slot_3: int: integer Range: 0 to 31
- Slot_4: int: integer Range: 0 to 31
- Slot_5: int: integer Range: 0 to 31
- Slot_6: int: integer Range: 0 to 31
- Slot_7: int: integer Range: 0 to 31

get_ab_search() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:ABSearch
value: bool = driver.configure.multiEval.get_ab_search()
```

Enables or disables the access burst measurement.

return enable: OFF | ON ON: Enable access burst search OFF: Disable access burst search

get_amode() → RsCmwGsmMeas.enums.AcquisitionMode

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:AMODe
value: enums.AcquisitionMode = driver.configure.multiEval.get_amode()
```

Selects the method that the R&S CMW uses for frame synchronization.

return acquisition_mode: GAP | PATTern GAP: Gap PATTern: Pattern

get_apattern() → ApatternStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:APATtern
value: ApatternStruct = driver.configure.multiEval.get_apattern()
```

Defines the burst pattern that the R&S CMW expects in the TDMA frames of the received GSM signal. The pattern is used for frame synchronization if the pattern acquisition mode is active (see method RsCmwGsmMeas.Configure.MultiEval.amode) .

return structure: for return value, see the help for ApatternStruct structure arguments.

get_fc_range() → RsCmwGsmMeas.enums.RangeMode

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:FCRange
value: enums.RangeMode = driver.configure.multiEval.get_fc_range()
```

Selects the width of the frequency range that the R&S CMW analyzes to establish time-synchronization with the received signal.

return mode: NORMal | WIDE NORMal: Normal frequency range WIDE: Wide frequency range

get_glength() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:GLENgth
value: int = driver.configure.multiEval.get_glength()
```

Defines the gap length as an integer number of slots. The gap length is used for frame synchronization if the gap acquisition mode is active (see method RsCmwGsmMeas.Configure.MultiEval.amode) .

return gap_length: integer Range: 1 slot to 3 slots, Unit: slots

get_hda_level() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:HDALevel
value: float or bool = driver.configure.multiEval.get_hda_level()
```

Defines a signal level relative to the 'Expected Nominal Power' (method RsCmwGsmMeas.Configure.RfSettings.envelopePower) where the two results obtained in a two stage measurement are joined.

return high_dyn_ass_level: numeric | ON | OFF Range: -60 dB to -10 dB, Unit: dB
Additional parameters: OFF | ON (disables | enables two-shot measurement)

get_iio_frames() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:IIOFrames
value: bool = driver.configure.multiEval.get_iio_frames()
```

Enables feature ignore initial off frames to avoid trigger timeout in access burst measurement in idle mode.

return ignore: OFF | ON

get_mo_exception() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MOEXception
value: bool = driver.configure.multiEval.get_mo_exception()
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected.

return meas_on_exception: ON | OFF ON: Results are never rejected OFF: Faulty results are rejected

get_mslots() → MslotsStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MSLots
value: MslotsStruct = driver.configure.multiEval.get_mslots()
```

Defines settings for the measured slots. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:MSLot:UL.

return structure: for return value, see the help for MslotsStruct structure arguments.

get_mview() → List[RsCmwGsmMeas.enums.SlotA]

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MVIew
value: List[enums.SlotA] = driver.configure.multiEval.get_mview()
```

Defines the expected modulation scheme and burst type in all timeslots and adjusts the power/time template accordingly.

return slot: No help available

get_nbq_search() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:NBQSearch
value: bool = driver.configure.multiEval.get_nbq_search()
```

Enables or disables the search for 16-QAM-modulated normal bursts.

return enable: OFF | ON ON: Enable 16-QAM NB search OFF: Disable 16-QAM NB search

get_pcl() → PclStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:PCL
value: PclStruct = driver.configure.multiEval.get_pcl()
```

Sets the expected PCL values in all timeslots, to be used in method RsCmwGsmMeas.Configure.MultiEval.pclModePCL. The PCL values are interpreted according to the current GSM band setting (method RsCmwGsmMeas.Configure.band) .

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:PCL:TCH:CSWitched
- CONFigure:GSM:SIGN<i>:RFSettings:CHCCombined:TCH:CSWitched

return structure: for return value, see the help for PclStruct structure arguments.

get_pcl_mode() → RsCmwGsmMeas.enums.PclMode

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:PCLMode
value: enums.PclMode = driver.configure.multiEval.get_pcl_mode()
```

Defines how the R&S CMW determines the PCL of the measured signal.

return pcl_mode: AUTO | PCL | SIGNaling AUTO: Estimated PCL PCL: PCL defined via method RsCmwGsmMeas.Configure.MultiEval.pcl SIGNaling: PCL determined by coupled signaling application (combined signal path only)

get_repetition() → RsCmwGsmMeas.enums.Repeat

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:REPetition
value: enums.Repeat = driver.configure.multiEval.get_repetition()
```

Specifies the repetition mode of the measurement. The repetition mode specifies whether the measurement is stopped after a single shot or repeated continuously. Use CONFigure:..:MEAS<i>:...:SCOunt to determine the number of measurement intervals per single shot.

return repetition: SINGleshot | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

get_rp_mode() → RsCmwGsmMeas.enums.RefPowerMode

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RPMode
value: enums.RefPowerMode = driver.configure.multiEval.get_rp_mode()
```

Defines how the reference power, i.e. the 0-dB line in the measurement diagram, is calculated.

return ref_power_mode: CURRent | DCOMpensated | AVERage Current, data compensated, average

get_scondition() → RsCmwGsmMeas.enums.StopCondition

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCONdition
value: enums.StopCondition = driver.configure.multiEval.get_scondition()
```

Qualifies whether the measurement is stopped after a failed limit check or continued. SLFail means that the measurement is stopped and reaches the RDY state when one of the results exceeds the limits.

return stop_condition: NONE | SLFail NONE: Continue measurement irrespective of the limit check SLFail: Stop measurement on limit failure

get_timeout() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:TOUT
value: float = driver.configure.multiEval.get_timeout()
```

Defines a timeout for the measurement. The timer is started when the measurement is initiated via a READ or INIT command. It is not started if the measurement is initiated manually ([ON | OFF] key or [RESTART | STOP] key) . When the measurement has completed the first measurement cycle (first single shot) , the statistical depth is reached and the timer is reset. If the first measurement cycle has not been completed when the timer expires, the measurement is stopped. The measurement state changes to RDY. The reliability indicator is set to 1, indicating that a measurement timeout occurred. Still running READ, FETCh or CALCulate commands are completed, returning the available results. At least for some results, there are no values at all or the statistical depth has not been reached. A timeout of 0 s corresponds to an infinite measurement timeout.

return timeout: numeric Unit: s

get_tsequence() → RsCmwGsmMeas.enums.TscA

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:TSEQuence
value: enums.TscA = driver.configure.multiEval.get_tsequence()
```

Selects the training sequence of the analyzed bursts. For the combined signal path scenario, use CONFigure:GSM:SIGN<i>:CELL:BCC.

return tsc: OFF | TSC0 | TSC1 | TSC2 | TSC3 | TSC4 | TSC5 | TSC6 | TSC7 | TSCA | DUMM OFF: Analyze all bursts, irrespective of their training sequence TSC0 ... TSC7:Analyze bursts with a particular GSM training sequence TSCA: Analyze bursts with any of the GSM training sequences TSC0 to TSC7 DUMMY: Analyze GSM-specific dummy bursts

set_ab_search(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:ABSearch
driver.configure.multiEval.set_ab_search(enable = False)
```

Enables or disables the access burst measurement.

param enable OFF | ON ON: Enable access burst search OFF: Disable access burst search

set_amode(*acquisition_mode: RsCmwGsmMeas.enums.AcquisitionMode*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:AMODe
driver.configure.multiEval.set_amode(acquisition_mode = enums.AcquisitionMode.
↪GAP)
```

Selects the method that the R&S CMW uses for frame synchronization.

param acquisition_mode GAP | PATTern GAP: Gap PATTern: Pattern

set_apattern(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval.MultiEval.ApatternStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:APATtern
driver.configure.multiEval.set_apattern(value = ApatternStruct())
```

Defines the burst pattern that the R&S CMW expects in the TDMA frames of the received GSM signal. The pattern is used for frame synchronization if the pattern acquisition mode is active (see method RsCmwGsmMeas.Configure.MultiEval.amode) .

param value see the help for ApatternStruct structure arguments.

set_fc_range(*mode: RsCmwGsmMeas.enums.RangeMode*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:FCRange
driver.configure.multiEval.set_fc_range(mode = enums.RangeMode.NORMal)
```

Selects the width of the frequency range that the R&S CMW analyzes to establish time-synchronization with the received signal.

param mode NORMal | WIDE NORMal: Normal frequency range WIDE: Wide frequency range

set_glength(*gap_length: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:GLENgth
driver.configure.multiEval.set_glength(gap_length = 1)
```

Defines the gap length as an integer number of slots. The gap length is used for frame synchronization if the gap acquisition mode is active (see method RsCmwGsmMeas.Configure.MultiEval.amode) .

param gap_length integer Range: 1 slot to 3 slots, Unit: slots

set_hda_level(*high_dyn_ass_level: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:HDALevel
driver.configure.multiEval.set_hda_level(high_dyn_ass_level = 1.0)
```

Defines a signal level relative to the 'Expected Nominal Power' (method RsCmwGsmMeas.Configure.RfSettings.envelopePower) where the two results obtained in a two stage measurement are joined.

param high_dyn_ass_level numeric | ON | OFF Range: -60 dB to -10 dB, Unit: dB Additional parameters: OFF | ON (disables | enables two-shot measurement)

set_iio_frames(*ignore: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:IIOFrames
driver.configure.multiEval.set_iio_frames(ignore = False)
```

Enables feature ignore initial off frames to avoid trigger timeout in access burst measurement in idle mode.

param ignore OFF | ON

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MOEXception
driver.configure.multiEval.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the R&S CMW identifies as faulty or inaccurate are rejected.

param meas_on_exception ON | OFF ON: Results are never rejected OFF: Faulty results are rejected

set_mslots(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval.MultiEval.MslotsStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MSLots
driver.configure.multiEval.set_mslots(value = MslotsStruct())
```

Defines settings for the measured slots. For the combined signal path scenario, useCONFigure:GSM:SIGN<i>:MSLot:UL.

param value see the help for MslotsStruct structure arguments.

set_mview(*slot: List[RsCmwGsmMeas.enums.SlotA]*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MVIew
driver.configure.multiEval.set_mview(slot = [SlotA.ACCess, SlotA.Q16])
```

Defines the expected modulation scheme and burst type in all timeslots and adjusts the power/time template accordingly.

param slot ANY | OFF | GMSK | EPSK | ACCess | Q16 ANY: Any burst type can be analyzed OFF: No signal expected GMSK: GMSK-modulated normal bursts EPSK: 8PSK-modulated normal bursts ACCess: Access bursts Q16: 16-QAM-modulated normal bursts

set_nbq_search(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:NBQSearch
driver.configure.multiEval.set_nbq_search(enable = False)
```

Enables or disables the search for 16-QAM-modulated normal bursts.

param enable OFF | ON ON: Enable 16-QAM NB search OFF: Disable 16-QAM NB search

set_pcl(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval.MultiEval.PclStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:PCL
driver.configure.multiEval.set_pcl(value = PclStruct())
```

Sets the expected PCL values in all timeslots, to be used in method RsCmwGsmMeas.Configure.MultiEval.pclModePCL. The PCL values are interpreted according to the current GSM band setting (method RsCmwGsmMeas.Configure.band) .

INTRO_CMD_HELP: For the combined signal path scenario, use:

- CONFigure:GSM:SIGN<i>:RFSettings:PCL:TCH:CSWitched
- CONFigure:GSM:SIGN<i>:RFSettings:CHCCombined:TCH:CSWitched

param value see the help for PclStruct structure arguments.

set_pcl_mode(*pcl_mode: RsCmwGsmMeas.enums.PclMode*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:PCLMode
driver.configure.multiEval.set_pcl_mode(pcl_mode = enums.PclMode.AUTO)
```

Defines how the R&S CMW determines the PCL of the measured signal.

param pcl_mode AUTO | PCL | SIGNaling AUTO: Estimated PCL PCL: PCL defined via method RsCmwGsmMeas.Configure.MultiEval.pcl SIGNaling: PCL determined by coupled signaling application (combined signal path only)

set_repetition(*repetition: RsCmwGsmMeas.enums.Repeat*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:REPetition
driver.configure.multiEval.set_repetition(repetition = enums.Repeat.CONTinuous)
```

Specifies the repetition mode of the measurement. The repetition mode specifies whether the measurement is stopped after a single shot or repeated continuously. Use CONFigure:..:MEAS<i>:...:SCOunt to determine the number of measurement intervals per single shot.

param repetition SINGleshot | CONTinuous SINGleshot: Single-shot measurement CONTinuous: Continuous measurement

set_rp_mode(*ref_power_mode: RsCmwGsmMeas.enums.RefPowerMode*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RPMode
driver.configure.multiEval.set_rp_mode(ref_power_mode = enums.RefPowerMode.
↪AVERage)
```

Defines how the reference power, i.e. the 0-dB line in the measurement diagram, is calculated.

param ref_power_mode CURRent | DCOMpensated | AVERage Current, data compensated, average

set_scondition(*stop_condition: RsCmwGsmMeas.enums.StopCondition*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCONdition
driver.configure.multiEval.set_scondition(stop_condition = enums.StopCondition.
↪NONE)
```

Qualifies whether the measurement is stopped after a failed limit check or continued. SLFail means that the measurement is stopped and reaches the RDY state when one of the results exceeds the limits.

param stop_condition NONE | SLFail NONE: Continue measurement irrespective of the limit check SLFail: Stop measurement on limit failure

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:TOUT
driver.configure.multiEval.set_timeout(timeout = 1.0)
```

Defines a timeout for the measurement. The timer is started when the measurement is initiated via a READ or INIT command. It is not started if the measurement is initiated manually ([ON | OFF] key or [RESTART | STOP] key) . When the measurement has completed the first measurement cycle (first single shot) , the statistical depth is reached and the timer is reset. If the first measurement cycle has not been completed when the timer expires, the measurement is stopped. The measurement state changes to RDY. The reliability indicator is set to 1, indicating that a measurement timeout occurred. Still running READ, FETCh or CALCulate commands are completed, returning the available results. At least for some results, there are no values at all or the statistical depth has not been reached. A timeout of 0 s corresponds to an infinite measurement timeout.

param timeout numeric Unit: s

set_tsequence(*tsc: RsCmwGsmMeas.enums.TscA*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:TSEQuence
driver.configure.multiEval.set_tsequence(tsc = enums.TscA.DUMM)
```

Selects the training sequence of the analyzed bursts. For the combined signal path scenario, use CONFigure:GSM:SIGN<i>:CELL:BCC.

param tsc OFF | TSC0 | TSC1 | TSC2 | TSC3 | TSC4 | TSC5 | TSC6 | TSC7 | TSCA | DUMM OFF: Analyze all bursts, irrespective of their training sequence TSC0 ... TSC7:Analyze bursts with a particular GSM training sequence TSCA: Analyze bursts with any of the GSM training sequences TSC0 to TSC7 DUMMY: Analyze GSM-specific dummy bursts

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.clone()
```

Subgroups

7.2.2.1 ListPy

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SLENgth
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:LRANge
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:OSINdex
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:IIFRames
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST
```

class ListPy

ListPy commands group definition. 13 total commands, 2 Sub-groups, 5 group commands

class LrangeStruct

Structure for reading output parameters. Fields:

- Start_Index: int: numeric First measured segment in the range of configured segments Range: 1 to 2000
- Nr_Segments: int: numeric Relative number within the range of measured segments Range: 1 to 512

get_ii_frames() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:IIFRames
value: bool = driver.configure.multiEval.listPy.get_ii_frames()
```

Selects whether idle frames are ignored or cause a 'signal low' error. For details, see 'Idle frame evaluation'.

return ignore: OFF | ON

get_lrange() → LrangeStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:LRANge
value: LrangeStruct = driver.configure.multiEval.listPy.get_lrange()
```

Select a range of measured segments. The segments must be configured using method RsCmwGsmMeas.Configure.MultiEval.ListPy. Segment.Setup.set.

return structure: for return value, see the help for LrangeStruct structure arguments.

get_os_index() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:OSINdex
value: int or bool = driver.configure.multiEval.listPy.get_os_index()
```

Selects the number of the segment to be displayed in the measurement diagram. The selected index must be within the range of measured segments (method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange) . Setting a value also enables the offline mode.

return offline_seg_index: numeric | ON | OFF Range: 1 to 200 Additional parameters: ON | OFF (enables | disables offline mode)

get_slength() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SLENgth
value: int or bool = driver.configure.multiEval.listPy.get_slength()
```

Selects the step length, i.e. the time difference between two measured TDMA timeslots. A step length of 1 means that every slot is measured, a step length of 8 means that a single timeslot per TDMA frame is measured.

INTRO_CMD_HELP: If the step length is set to OFF, an arbitrary number of slots in each TDMA frame can be measured. The measured slots are defined by the <FramePattern> parameter of the following commands:

- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Modulation.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.PowerVsTime.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Smodulation.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Sswitching.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Ber.set

return step_length: numeric | ON | OFF Step length as number of TDMA slots Range: 1 to 8 Additional parameters: ON | OFF (enable step length | use FramePattern)

get_value() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST
value: bool = driver.configure.multiEval.listPy.get_value()
```

Enables or disables the list mode.

return enable: OFF | ON ON: Enable list mode OFF: Disable list mode

set_ii_frames(*ignore: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:IIFRames
driver.configure.multiEval.listPy.set_ii_frames(ignore = False)
```

Selects whether idle frames are ignored or cause a 'signal low' error. For details, see 'Idle frame evaluation'.

param ignore OFF | ON

set_lrange(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.ListPy.ListPy.LrangeStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:LRANge
driver.configure.multiEval.listPy.set_lrange(value = LrangeStruct())
```

Select a range of measured segments. The segments must be configured using method RsCmwGsmMeas.Configure.MultiEval.ListPy. Segment.Setup.set.

param value see the help for LrangeStruct structure arguments.

set_os_index(*offline_seg_index: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:OSINdex
driver.configure.multiEval.listPy.set_os_index(offline_seg_index = 1)
```

Selects the number of the segment to be displayed in the measurement diagram. The selected index must be within the range of measured segments (method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange) . Setting a value also enables the offline mode.

param offline_seg_index numeric | ON | OFF Range: 1 to 200 Additional parameters: ON | OFF (enables | disables offline mode)

set_slength(*step_length: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SLENgth
driver.configure.multiEval.listPy.set_slength(step_length = 1)
```

Selects the step length, i.e. the time difference between two measured TDMA timeslots. A step length of 1 means that every slot is measured, a step length of 8 means that a single timeslot per TDMA frame is measured.

INTRO_CMD_HELP: If the step length is set to OFF, an arbitrary number of slots in each TDMA frame can be measured. The measured slots are defined by the <FramePattern> parameter of the following commands:

- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Modulation.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.PowerVsTime.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Smodulation.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Sswitching.set
- method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.Ber.set

param step_length numeric | ON | OFF Step length as number of TDMA slots Range: 1 to 8 Additional parameters: ON | OFF (enable step length | use FramePattern)

set_value(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST
driver.configure.multiEval.listPy.set_value(enable = False)
```

Enables or disables the list mode.

param enable OFF | ON ON: Enable list mode OFF: Disable list mode

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.listPy.clone()
```

Subgroups

7.2.2.1.1 Segment

RepCap Settings

```
# Range: Nr1 .. Nr512
rc = driver.configure.multiEval.listPy.segment.repcap_segment_get()
driver.configure.multiEval.listPy.segment.repcap_segment_set(repcap.Segment.Nr1)
```

class Segment

Segment commands group definition. 7 total commands, 7 Sub-groups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.listPy.segment.clone()
```

Subgroups

7.2.2.1.1.1 Setup

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:SETup
```

class Setup

Setup commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class SetupStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Segment_Length: int: integer Number of steps or frames in the segment, depending on the configured step length ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:LIST:SLENgth CMDLINK]) . If the step length is set to OFF, the segment length is defined in frames. So the number of slots in the segment equals 8 * SegmentLength. If a step length is defined (1 to 8) , the segment length is defined in steps. So the number of slots in the segment equals StepLength * SegmentLength. Range: 1 to 3000
- Level: float: numeric Expected nominal power in the segment. The range of the expected nominal power can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin The input power range is stated in the data sheet. Unit: dBm
- Frequency: float: numeric Range: 100 MHz to 6 GHz, Unit: Hz
- Pcl: int: Optional setting parameter. integer Expected power control level for the segment Range: 0 to 31
- Retrigger_Flag: enums.RetriggerFlag: Optional setting parameter. OFF | ON | IFPower Specifies whether a trigger event is required for the segment or not. The setting is ignored for the first segment of a measurement and for trigger mode ONCE (see [CMDLINK: TRIGger:GSM:MEASi:MEValuation:LIST:MODE CMDLINK]) . OFF: measure the segment without retrigger ON: wait for trigger event before measuring the segment IFPower: wait for 'IF Power' trigger event before measuring the segment

- Evaluat_Offset: int: Optional setting parameter. integer Number of steps at the beginning of the segment which are not measured Range: 0 to 1000

get(*segment=<Segment.Default: -1>*) → SetupStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SETup
value: SetupStruct = driver.configure.multiEval.listPy.segment.setup.
↪get(segment = repcap.Segment.Default)
```

Defines the length, the analyzer settings, the expected PCL, retrigger setting and evaluation offset of a selected segment. In general, this command must be sent for all measured segments (method RsCmwGsmMeas.Configure.MultiEval.ListPy. lrange) . The PCL values are used if the global 'PCL Mode: PCL' is set (method RsCmwGsmMeas.Configure.MultiEval. pclModePCL) . They can affect the limit check results; see 'PCL Mode'. The current GSM band setting (method RsCmwGsmMeas. Configure.band) specifies the exact meaning of the PCL; see Table 'GSM power control levels'.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for SetupStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.ListPy_.Segment_.Setup.Setup.SetupStruct*, *segment=<Segment.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SETup
driver.configure.multiEval.listPy.segment.setup.set(value = [PROPERTY_STRUCT_
↪NAME](), segment = repcap.Segment.Default)
```

Defines the length, the analyzer settings, the expected PCL, retrigger setting and evaluation offset of a selected segment. In general, this command must be sent for all measured segments (method RsCmwGsmMeas.Configure.MultiEval.ListPy. lrange) . The PCL values are used if the global 'PCL Mode: PCL' is set (method RsCmwGsmMeas.Configure.MultiEval. pclModePCL) . They can affect the limit check results; see 'PCL Mode'. The current GSM band setting (method RsCmwGsmMeas. Configure.band) specifies the exact meaning of the PCL; see Table 'GSM power control levels'.

param structure for set value, see the help for SetupStruct structure arguments.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

7.2.2.1.1.2 Modulation

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation
```

class Modulation

Modulation commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class ModulationStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Mod_Statistics: int: integer The statistical length is defined in slots. It is limited by the number of evaluated slots (defined via step length or frame pattern) . Range: 1 to 1000

- Evm_Enable: bool: OFF | ON ON: Enable measurement of EVM OFF: Disable measurement of EVM
- Mag_Error_Enable: bool: OFF | ON Enable or disable measurement of magnitude error
- Phase_Err_Enable: bool: OFF | ON Enable or disable measurement of phase error
- Am_Pm_Enable: bool: OFF | ON Enable or disable measurement of AM PM delay
- Frame_Pattern: str: Optional setting parameter. binary 8-digit binary value, defines the evaluated timeslots in each TDMA frame. Used only if no step length is configured (see [CMDLINK: CONFigure:GSM:MEASi:MEValuation:LIST:SLENgth CMDLINK]) . Range: #B00000000 to #B11111111 (no slots ... all slots measured)

get(*segment=<Segment.Default: -1>*) → ModulationStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation
value: ModulationStruct = driver.configure.multiEval.listPy.segment.modulation.
↪get(segment = repcap.Segment.Default)
```

Defines the statistical length for the AVERage, MIN, and MAX calculation and enables the calculation of the different modulation results in segment no. <no>; see 'List Mode'.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for ModulationStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.ListPy_.Segment_.Modulation.Modulation.ModulationStruct*, *segment=<Segment.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation
driver.configure.multiEval.listPy.segment.modulation.set(value = [PROPERTY_
↪STRUCT_NAME](), segment = repcap.Segment.Default)
```

Defines the statistical length for the AVERage, MIN, and MAX calculation and enables the calculation of the different modulation results in segment no. <no>; see 'List Mode'.

param structure for set value, see the help for ModulationStruct structure arguments.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

7.2.2.1.1.3 PowerVsTime

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime
```

class PowerVsTime

PowerVsTime commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class PowerVsTimeStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Statistic: int: integer The statistical length is defined in slots. It is limited by the number of evaluated slots (defined via step length or frame pattern) . Range: 1 to 1000
- Enable: bool: OFF | ON ON: Enable measurement of power vs. time OFF: Disable measurement
- Frame_Pattern: str: Optional setting parameter. binary 8-digit binary value, defines the evaluated timeslots in each TDMA frame. Used only if no step length is configured (see [CMDLINK: CONFigure:GSM:MEASi:MEValuation:LIST:SLENgth CMDLINK]) . Range: #B00000000 to #B11111111 (no slots ... all slots measured)

get(*segment=<Segment.Default: -1>*) → PowerVsTimeStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PVTime
value: PowerVsTimeStruct = driver.configure.multiEval.listPy.segment.
↪powerVsTime.get(segment = repcap.Segment.Default)
```

Defines the statistical length for averaging and enables the power vs. time measurement in segment no. <no>; see 'List Mode'.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for PowerVsTimeStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.ListPy_.Segment_.PowerVsTime.PowerVsTime.PowerVsTimeStruct*, *segment=<Segment.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PVTime
driver.configure.multiEval.listPy.segment.powerVsTime.set(value = [PROPERTY_
↪STRUCT_NAME](), segment = repcap.Segment.Default)
```

Defines the statistical length for averaging and enables the power vs. time measurement in segment no. <no>; see 'List Mode'.

param structure for set value, see the help for PowerVsTimeStruct structure arguments.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

7.2.2.1.1.4 Smodulation

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:SMODulation
```

class Smodulation

Smodulation commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class SmodulationStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Statistic: int: integer The statistical length is defined in slots. It is limited by the number of evaluated slots (defined via step length or frame pattern) . Range: 1 to 1000

- Enable: bool: OFF | ON ON: Enable measurement of spectrum due to modulation results (including the 'spectrum modulation time' results in offline mode) OFF: Disable measurement
- Frame_Pattern: str: Optional setting parameter. binary 8-digit binary value, defines the evaluated timeslots in each TDMA frame. Used only if no step length is configured (see [CMDLINK: CONFigure:GSM:MEASi:MEValuation:LIST:SLENgth CMDLINK]) . Range: #B00000000 to #B11111111 (no slots … all slots measured)

get(*segment=<Segment.Default: -1>*) → SmodulationStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SMODulation
value: SmodulationStruct = driver.configure.multiEval.listPy.segment.
↪smodulation.get(segment = repcap.Segment.Default)
```

Defines the statistical length for averaging and enables the spectrum due to modulation measurement in segment no. <no>; see 'List Mode'.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for SmodulationStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.ListPy_.Segment_.Smodulation.Smodulation.SmodulationStruct*, *segment=<Segment.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SMODulation
driver.configure.multiEval.listPy.segment.smodulation.set(value = [PROPERTY_
↪STRUCT_NAME](), segment = repcap.Segment.Default)
```

Defines the statistical length for averaging and enables the spectrum due to modulation measurement in segment no. <no>; see 'List Mode'.

param structure for set value, see the help for SmodulationStruct structure arguments.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

7.2.2.1.1.5 Sswitching

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:SSWitching
```

class Sswitching

Sswitching commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class SswitchingStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Statistic: int: integer The statistical length is defined in slots. It is limited by the number of evaluated slots (defined via step length or frame pattern) . Range: 1 to 100

- Enable: bool: OFF | ON ON: Enable measurement of spectrum due to switching (including the 'spectrum switching time' results in offline mode) OFF: Disable measurement
- Frame_Pattern: str: Optional setting parameter. binary 8-digit binary value, defines the evaluated timeslots in each TDMA frame. Used only if no step length is configured (see [CMDLINK: CONFigure:GSM:MEASi:MEValuation:LIST:SLENgth CMDLINK]) . Range: #B00000000 to #B11111111 (no slots … all slots measured)

get(*segment=<Segment.Default: -1>*) → SswitchingStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SSWitching
value: SswitchingStruct = driver.configure.multiEval.listPy.segment.sswitching.
↪get(segment = repcap.Segment.Default)
```

Defines the statistical length for the maximum calculation (peak hold mode) and enables the spectrum due to switching measurement in segment no. <no>; see 'List Mode'.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for SswitchingStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.ListPy_.Segment_.Sswitching.Sswitching.SswitchingStruct*, *segment=<Segment.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SSWitching
driver.configure.multiEval.listPy.segment.sswitching.set(value = [PROPERTY_
↪STRUCT_NAME](), segment = repcap.Segment.Default)
```

Defines the statistical length for the maximum calculation (peak hold mode) and enables the spectrum due to switching measurement in segment no. <no>; see 'List Mode'.

param structure for set value, see the help for SswitchingStruct structure arguments.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

7.2.2.1.1.6 Ber

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:BER
```

class Ber

Ber commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class BerStruct

Structure for setting input parameters. Contains optional setting parameters. Fields:

- Statistic: int: integer The statistical length is defined in slots. It is limited by the number of evaluated slots (defined via step length or frame pattern) . Range: 1 to 100
- Enable: bool: OFF | ON ON: Enable BER measurement OFF: Disable measurement

- Loop_Type: enums.LoopType: C | SRB C: Loop C SRB: SRB loop
- Frame_Pattern: str: Optional setting parameter. binary 8-digit binary value, defines the evaluated timeslots in each TDMA frame. Used only if no step length is configured (see [CMDLINK: CONFigure:GSM:MEASi:MEValuation:LIST:SLENgth CMDLINK]) . Range: #B00000000 to #B11111111 (no slots … all slots measured)

get(*segment=<Segment.Default: -1>*) → BerStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:BER
value: BerStruct = driver.configure.multiEval.listPy.segment.ber.get(segment =␣
↪repcap.Segment.Default)
```

Defines the statistical length for averaging and enables the BER measurement in segment no. <no>; see 'List Mode'.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for BerStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.ListPy_.Segment_.Ber.Ber.BerStruct*, *segment=<Segment.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:BER
driver.configure.multiEval.listPy.segment.ber.set(value = [PROPERTY_STRUCT_
↪NAME](), segment = repcap.Segment.Default)
```

Defines the statistical length for averaging and enables the BER measurement in segment no. <no>; see 'List Mode'.

param structure for set value, see the help for BerStruct structure arguments.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

7.2.2.1.1.7 SingleCmw

class SingleCmw

SingleCmw commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.listPy.segment.singleCmw.clone()
```

Subgroups

7.2.2.1.1.8 Connector

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:CMWS:CONNector
```

class Connector

Connector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*segment*=<*Segment.Default: -1*>) → RsCmwGsmMeas.enums.CmwsConnector

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CMWS:CONNector
value: enums.CmwsConnector = driver.configure.multiEval.listPy.segment.
↪singleCmw.connector.get(segment = repcap.Segment.Default)
```

Selects the RF input connector for segment <no> for GSM list mode measurements with the R&S CMWS. This setting is only relevant for connector mode LIST, see method RsCmwGsmMeas.Configure.MultiEval.ListPy.SingleCmw.cmode. All segments of a list mode measurement must use connectors of the same bench. For possible connector values, see 'Values for RF Path Selection'.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return cmws_connector: Selects the input connector of the R&S CMWS

set(*cmws_connector: RsCmwGsmMeas.enums.CmwsConnector*, *segment*=<*Segment.Default: -1*>) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CMWS:CONNector
driver.configure.multiEval.listPy.segment.singleCmw.connector.set(cmws_
↪connector = enums.CmwsConnector.R11, segment = repcap.Segment.Default)
```

Selects the RF input connector for segment <no> for GSM list mode measurements with the R&S CMWS. This setting is only relevant for connector mode LIST, see method RsCmwGsmMeas.Configure.MultiEval.ListPy.SingleCmw.cmode. All segments of a list mode measurement must use connectors of the same bench. For possible connector values, see 'Values for RF Path Selection'.

param cmws_connector Selects the input connector of the R&S CMWS

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

7.2.2.1.2 SingleCmw

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:CMWS:CMODe
```

class SingleCmw

SingleCmw commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_cmode() → RsCmwGsmMeas.enums.ParameterSetMode

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:CMWS:CMODe
value: enums.ParameterSetMode = driver.configure.multiEval.listPy.singleCmw.get_
↪cmode()
```

Specifies how the input connector is selected for GSM list mode measurements with the R&S CMWS.

return connector_mode: GLOBal | LIST GLOBal: The same input connector is used for all segments. It is selected in the same way as without list mode, for example via ROUTe:GSM:MEASi:SCENario:SALone. LIST: The input connector is configured individually for each segment. See method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.SingleCmw.Connector.set.

set_cmode(*connector_mode: RsCmwGsmMeas.enums.ParameterSetMode*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:CMWS:CMODe
driver.configure.multiEval.listPy.singleCmw.set_cmode(connector_mode = enums.
↪ParameterSetMode.GLOBal)
```

Specifies how the input connector is selected for GSM list mode measurements with the R&S CMWS.

param connector_mode GLOBal | LIST GLOBal: The same input connector is used for all segments. It is selected in the same way as without list mode, for example via ROUTe:GSM:MEASi:SCENario:SALone. LIST: The input connector is configured individually for each segment. See method RsCmwGsmMeas.Configure.MultiEval.ListPy.Segment.SingleCmw.Connector.set.

7.2.2.2 Vamos

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:VAMos:TSCSet
```

class Vamos

Vamos commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_tsc_set() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:VAMos:TSCSet
value: int = driver.configure.multiEval.vamos.get_tsc_set()
```

Specifies the expected VAMOS training sequence code (TSC) set of the measured GSM uplink signal. With a specific TSC set selection, the R&S CMW analyzes bursts with this TSC set only.

return tsc_set: integer Range: 1 to 2

set_tsc_set(*tsc_set: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:VAMos:TSCSet
driver.configure.multiEval.vamos.set_tsc_set(tsc_set = 1)
```

Specifies the expected VAMOS training sequence code (TSC) set of the measured GSM uplink signal. With a specific TSC set selection, the R&S CMW analyzes bursts with this TSC set only.

param tsc_set integer Range: 1 to 2

7.2.2.3 FilterPy

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:FILTer:PVTime
CONFigure:GSM:MEASurement<Instance>:MEValuation:FILTer:IQ
```

class FilterPy

FilterPy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_iq() → RsCmwGsmMeas.enums.FilterIq

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:FILTer:IQ
value: enums.FilterIq = driver.configure.multiEval.filterPy.get_iq()
```

Specifies whether the I/Q data is filtered to eliminate the inter-symbol interference (ISI) at all constellation points.

return filter_py: ISIRemoved | UNFiltered | F90Khz ISIRemoved: ISI removed UNFiltered: Unfiltered data F90Khz: 90 kHz filter

get_power_vs_time() → RsCmwGsmMeas.enums.FilterPvTime

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:FILTer:PVTime
value: enums.FilterPvTime = driver.configure.multiEval.filterPy.get_power_vs_
↪time()
```

Selects the bandwidth of the IF filter.

return filter_py: G05M | G10M G05M: 500 kHz Gauss filter G10M: 1 MHz Gauss filter

set_iq(*filter_py: RsCmwGsmMeas.enums.FilterIq*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:FILTer:IQ
driver.configure.multiEval.filterPy.set_iq(filter_py = enums.FilterIq.F90Khz)
```

Specifies whether the I/Q data is filtered to eliminate the inter-symbol interference (ISI) at all constellation points.

param filter_py ISIRemoved | UNFiltered | F90Khz ISIRemoved: ISI removed UNFiltered: Unfiltered data F90Khz: 90 kHz filter

set_power_vs_time(*filter_py: RsCmwGsmMeas.enums.FilterPvTime*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:FILTer:PVTime
driver.configure.multiEval.filterPy.set_power_vs_time(filter_py = enums.
↪FilterPvTime.G05M)
```

Selects the bandwidth of the IF filter.

param filter_py G05M | G10M G05M: 500 kHz Gauss filter G10M: 1 MHz Gauss filter

7.2.2.4 Rotation

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:ROTation:IQ
```

class Rotation

Rotation commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_iq() → RsCmwGsmMeas.enums.Rotation

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:ROTation:IQ
value: enums.Rotation = driver.configure.multiEval.rotation.get_iq()
```

Specifies whether the rotation of the 8PSK and 16-QAM symbols is subtracted off before the symbols are displayed in the constellation diagram.

return rotation: P38 | P38R P38: Rotation not removed, phase-rotated symbols displayed P38R: Rotation removed

set_iq(*rotation: RsCmwGsmMeas.enums.Rotation*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:ROTation:IQ
driver.configure.multiEval.rotation.set_iq(rotation = enums.Rotation.P38)
```

Specifies whether the rotation of the 8PSK and 16-QAM symbols is subtracted off before the symbols are displayed in the constellation diagram.

param rotation P38 | P38R P38: Rotation not removed, phase-rotated symbols displayed P38R: Rotation removed

7.2.2.5 Modulation

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:MODulation:DECode
```

class Modulation

Modulation commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_decode() → RsCmwGsmMeas.enums.Decode

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MODulation:DECode
value: enums.Decode = driver.configure.multiEval.modulation.get_decode()
```

Defines whether the guard or tail bits are decoded.

return decode: STANdard | GTBits STANdard: Guard and tail bits are assumed to be in line with GSM and therefore not decoded. GTBits: Guard and tail bits are also decoded.

set_decode(*decode: RsCmwGsmMeas.enums.Decode*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:MODulation:DECode
driver.configure.multiEval.modulation.set_decode(decode = enums.Decode.GTBits)
```

Defines whether the guard or tail bits are decoded.

param decode STANdard | GTBits STANdard: Guard and tail bits are assumed to be in line with GSM and therefore not decoded. GTBits: Guard and tail bits are also decoded.

7.2.2.6 Scount

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:PVTime
CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:MODulation
CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:SMODulation
CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:SSWitching
CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:BER
```

class Scount

Scount commands group definition. 5 total commands, 0 Sub-groups, 5 group commands

get_ber() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:BER
value: int = driver.configure.multiEval.scount.get_ber()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return statistic_count: numeric Number of measurement intervals (bursts) for the 'BER' measurement Range: 1 to 1000

get_modulation() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:MODulation
value: int = driver.configure.multiEval.scount.get_modulation()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return statistic_count: numeric Number of measurement intervals for the modulation measurement Range: 1 to 1000

get_power_vs_time() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:PVTime
value: int = driver.configure.multiEval.scount.get_power_vs_time()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return statistic_count: numeric Number of measurement intervals for the power vs. time measurement Range: 1 to 1000

get_smodulation() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:SMODulation
value: int = driver.configure.multiEval.scount.get_smodulation()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return statistic_count: numeric Number of measurement intervals for the spectrum modulation measurement Range: 1 to 1000

get_sswitching() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:SSWitching
value: int = driver.configure.multiEval.scount.get_sswitching()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

return statistic_count: numeric Number of measurement intervals for the spectrum switching measurement Range: 1 to 100

set_ber(*statistic_count: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:BER
driver.configure.multiEval.scount.set_ber(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count numeric Number of measurement intervals (bursts) for the 'BER' measurement Range: 1 to 1000

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:MODulation
driver.configure.multiEval.scount.set_modulation(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count numeric Number of measurement intervals for the modulation measurement Range: 1 to 1000

set_power_vs_time(*statistic_count: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:PVTime
driver.configure.multiEval.scount.set_power_vs_time(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count numeric Number of measurement intervals for the power vs. time measurement Range: 1 to 1000

set_smodulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:SMODulation
driver.configure.multiEval.scount.set_smodulation(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count numeric Number of measurement intervals for the spectrum modulation measurement Range: 1 to 1000

set_sswitching(*statistic_count: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SCOunt:SSWitching
driver.configure.multiEval.scount.set_sswitching(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot.

param statistic_count numeric Number of measurement intervals for the spectrum switching measurement Range: 1 to 100

7.2.2.7 Result

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:ALL
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:PVTime
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:MERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:PERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SMFRequency
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SMTime
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SSFRequency
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SSTime
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:AMPM
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:MSCalar
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:BER
CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:IQ
```

class Result

Result commands group definition. 13 total commands, 0 Sub-groups, 13 group commands

class AllStruct

Structure for reading output parameters. Fields:

- Power_Vs_Time: bool: OFF | ON Power vs. time ON: Evaluate results and show the view OFF: Do not evaluate results, hide the view (if applicable)
- Evm: bool: OFF | ON Error vector magnitude
- Magnitude_Error: bool: OFF | ON Magnitude error
- Phase_Error: bool: OFF | ON Phase error
- Iq: bool: OFF | ON I/Q constellation
- Acp_Mod_Frequency: bool: OFF | ON ACP spectrum modulation frequency
- Acp_Mod_Time: bool: OFF | ON ACP spectrum modulation time
- Acp_Swit_Freq: bool: OFF | ON ACP spectrum switching frequency
- Acp_Swit_Time: bool: OFF | ON ACP spectrum switching time
- Mod_Scalar: bool: OFF | ON Scalar modulation results
- Ber: bool: OFF | ON Bit error rate
- Am_Pm: bool: OFF | ON AM-PM

get_all() → AllStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult[:ALL]
value: AllStruct = driver.configure.multiEval.result.get_all()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. This command combines all other CONFigure:GSM:MEAS<i>:MEValuation:RESult... commands.

return structure: for return value, see the help for AllStruct structure arguments.

get_am_pm() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:AMPM
value: bool = driver.configure.multiEval.result.get_am_pm()
```

Enables or disables the evaluation of the AM-PM results, the scalar modulation results, and the bit error rate (BER) .

return enable: ON | OFF ON: Evaluate results OFF: Do not evaluate results

get_ber() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:BER
value: bool = driver.configure.multiEval.result.get_ber()
```

Enables or disables the evaluation of the AM-PM results, the scalar modulation results, and the bit error rate (BER) .

return enable: ON | OFF ON: Evaluate results OFF: Do not evaluate results

get_ev_magnitude() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
value: bool = driver.configure.multiEval.result.get_ev_magnitude()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_iq() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:IQ
value: bool = driver.configure.multiEval.result.get_iq()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_merror() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:MERRor
value: bool = driver.configure.multiEval.result.get_merror()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_mscalar() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:MSCalar
value: bool = driver.configure.multiEval.result.get_mscalar()
```

Enables or disables the evaluation of the AM-PM results, the scalar modulation results, and the bit error rate (BER) .

return enable: ON | OFF ON: Evaluate results OFF: Do not evaluate results

get_perror() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:PERRor
value: bool = driver.configure.multiEval.result.get_perror()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_power_vs_time() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:PVTime
value: bool = driver.configure.multiEval.result.get_power_vs_time()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_sm_frequency() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SMFRequency
value: bool = driver.configure.multiEval.result.get_sm_frequency()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_sm_time() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SMTime
value: bool = driver.configure.multiEval.result.get_sm_time()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_ss_frequency() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SSFRequency
value: bool = driver.configure.multiEval.result.get_ss_frequency()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

get_ss_time() → bool

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SSTime
value: bool = driver.configure.multiEval.result.get_ss_time()
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

return enable: ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_all(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Result.Result.AllStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult[:ALL]
driver.configure.multiEval.result.set_all(value = AllStruct())
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. This command combines all other CONFigure:GSM:MEAS<i>:MEValuation:RESult... commands.

param value see the help for AllStruct structure arguments.

set_am_pm(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:AMPM
driver.configure.multiEval.result.set_am_pm(enable = False)
```

Enables or disables the evaluation of the AM-PM results, the scalar modulation results, and the bit error rate (BER) .

param enable ON | OFF ON: Evaluate results OFF: Do not evaluate results

set_ber(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:BER
driver.configure.multiEval.result.set_ber(enable = False)
```

Enables or disables the evaluation of the AM-PM results, the scalar modulation results, and the bit error rate (BER) .

param enable ON | OFF ON: Evaluate results OFF: Do not evaluate results

set_ev_magnitude(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
driver.configure.multiEval.result.set_ev_magnitude(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_iq(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:IQ
driver.configure.multiEval.result.set_iq(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_merror(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:MERRor
driver.configure.multiEval.result.set_merror(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_mscalar(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:MSCalar
driver.configure.multiEval.result.set_mscalar(enable = False)
```

Enables or disables the evaluation of the AM-PM results, the scalar modulation results, and the bit error rate (BER) .

param enable ON | OFF ON: Evaluate results OFF: Do not evaluate results

set_perror(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:PERRor
driver.configure.multiEval.result.set_perror(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_power_vs_time(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:PVTime
driver.configure.multiEval.result.set_power_vs_time(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_sm_frequency(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SMFRequency
driver.configure.multiEval.result.set_sm_frequency(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_sm_time(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SMTime
driver.configure.multiEval.result.set_sm_time(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_ss_frequency(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SSFRequency
driver.configure.multiEval.result.set_ss_frequency(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

set_ss_time(*enable: bool*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:RESult:SSTime
driver.configure.multiEval.result.set_ss_time(enable = False)
```

Enables or disables the evaluation of results and shows or hides the views in the multi-evaluation measurement. The last mnemonic denotes the view type: Power vs. time, error vector magnitude, magnitude error, phase error, I/Q constellation, spectrum modulation frequency, spectrum modulation time, spectrum switching frequency, spectrum switching time. Use READ.. .? queries to retrieve results for disabled views.

param enable ON | OFF ON: Evaluate results and show view OFF: Do not evaluate results, hide view

7.2.2.8 Limit

class Limit

Limit commands group definition. 59 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.clone()
```

Subgroups

7.2.2.8.1 PowerVsTime

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:PVTime:GPLevel
```

class PowerVsTime

PowerVsTime commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

get_gp_level() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:PVTime:GPLevel
value: float or bool = driver.configure.multiEval.limit.powerVsTime.get_gp_
↪level()
```

Defines the raising of the upper limit line in the guard period between two consecutive bursts.

return guard_period_lev: numeric | ON | OFF Range: 0 dB to 10 dB, Unit: dB Additional parameters: OFF | ON (disables | enables the limit)

set_gp_level(*guard_period_lev: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:PVTime:GPLevel
driver.configure.multiEval.limit.powerVsTime.set_gp_level(guard_period_lev = 1.
↪0)
```

Defines the raising of the upper limit line in the guard period between two consecutive bursts.

param guard_period_lev numeric | ON | OFF Range: 0 dB to 10 dB, Unit: dB Additional parameters: OFF | ON (disables | enables the limit)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.powerVsTime.clone()
```

Subgroups

7.2.2.8.1.1 AbPower<AbPower>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.configure.multiEval.limit.powerVsTime.abPower.repcap_abPower_get()
driver.configure.multiEval.limit.powerVsTime.abPower.repcap_abPower_set(repcap.AbPower.
↪Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:PVTime:ABPower<AbPower>
```

class AbPower

AbPower commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: AbPower, default value after init: AbPower.Nr1

class AbPowerStruct

Structure for setting input parameters. Fields:

- Start_Pcl: int: integer Number of first TPCL to which the limits are applied Range: 0 to 31

- End_Pcl: int: integer Number of last TPCL to which the limits are applied Range: 0 to 31
- Lower_Limit: float: numeric Range: -10 dB to 0 dB, Unit: dB
- Upper_Limit: float: numeric Range: 0 dB to 10 dB, Unit: dB
- Enable: bool: OFF | ON ON: Enable limits for the given no OFF: Disable limits for the given no

get(*abPower=<AbPower.Default: -1>*) → AbPowerStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:PVTime:ABPower<nr>
value: AbPowerStruct = driver.configure.multiEval.limit.powerVsTime.abPower.
↪get(abPower = repcap.AbPower.Default)
```

Defines and activates limits for the average burst power, i.e. tolerances for ranges of template power control levels (TPCLs) .

param abPower optional repeated capability selector. Default value: Nr1 (settable in the interface 'AbPower')

return structure: for return value, see the help for AbPowerStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.PowerVsTime_.AbPower.AbPower.AbPowerStruct*, *abPower=<AbPower.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:PVTime:ABPower<nr>
driver.configure.multiEval.limit.powerVsTime.abPower.set(value = [PROPERTY_
↪STRUCT_NAME](), abPower = repcap.AbPower.Default)
```

Defines and activates limits for the average burst power, i.e. tolerances for ranges of template power control levels (TPCLs) .

param structure for set value, see the help for AbPowerStruct structure arguments.

param abPower optional repeated capability selector. Default value: Nr1 (settable in the interface 'AbPower')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.powerVsTime.abPower.clone()
```

7.2.2.8.2 Gmsk

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:EVMagnitude
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:MERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:IQOFfset
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:IQIMbalance
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:TERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:FERRor
```

class Gmsk
Gmsk commands group definition. 19 total commands, 3 Sub-groups, 7 group commands

class EvMagnitudeStruct
Structure for reading output parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

class FreqErrorStruct
Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

class IqImbalanceStruct
Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

class IqOffsetStruct
Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

class MerrorStruct
Structure for reading output parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

class PerrorStruct
Structure for reading output parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

class TerrorStruct
Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

get_ev_magnitude() → EvMagnitudeStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:EVMagnitude
value: EvMagnitudeStruct = driver.configure.multiEval.limit.gmsk.get_ev_
↪magnitude()
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the error vector magnitude (EVM) .

return structure: for return value, see the help for EvMagnitudeStruct structure arguments.

get_freq_error() → FreqErrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:FERRor
value: FreqErrorStruct = driver.configure.multiEval.limit.gmsk.get_freq_error()
```

Defines and activates upper limits for the frequency error.

return structure: for return value, see the help for FreqErrorStruct structure arguments.

get_iq_imbalance() → IqImbalanceStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:IQIMbalance
value: IqImbalanceStruct = driver.configure.multiEval.limit.gmsk.get_iq_
↪imbalance()
```

Defines and activates upper limits for the I/Q imbalance values.

return structure: for return value, see the help for IqImbalanceStruct structure arguments.

get_iq_offset() → IqOffsetStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:IQOFfset
value: IqOffsetStruct = driver.configure.multiEval.limit.gmsk.get_iq_offset()
```

Defines and activates upper limits for the I/Q origin offset values.

return structure: for return value, see the help for IqOffsetStruct structure arguments.

get_merror() → MerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:MERRor
value: MerrorStruct = driver.configure.multiEval.limit.gmsk.get_merror()
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the magnitude error.

return structure: for return value, see the help for MerrorStruct structure arguments.

get_perror() → PerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PERRor
value: PerrorStruct = driver.configure.multiEval.limit.gmsk.get_perror()
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the phase error.

return structure: for return value, see the help for PerrorStruct structure arguments.

get_terror() → TerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:TERRor
value: TerrorStruct = driver.configure.multiEval.limit.gmsk.get_terror()
```

Defines and activates upper limits for the timing error.

return structure: for return value, see the help for TerrorStruct structure arguments.

set_ev_magnitude(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk.Gmsk.EvMagnitudeStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:EVMagnitude
driver.configure.multiEval.limit.gmsk.set_ev_magnitude(value =␣
↪EvMagnitudeStruct())
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the error vector magnitude (EVM) .

param value see the help for EvMagnitudeStruct structure arguments.

set_freq_error(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk.Gmsk.FreqErrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:FERRor
driver.configure.multiEval.limit.gmsk.set_freq_error(value = FreqErrorStruct())
```

Defines and activates upper limits for the frequency error.

param value see the help for FreqErrorStruct structure arguments.

set_iq_imbalance(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk.Gmsk.IqImbalanceStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:IQIMbalance
driver.configure.multiEval.limit.gmsk.set_iq_imbalance(value =␣
↪IqImbalanceStruct())
```

Defines and activates upper limits for the I/Q imbalance values.

param value see the help for IqImbalanceStruct structure arguments.

set_iq_offset(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk.Gmsk.IqOffsetStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:IQOFfset
driver.configure.multiEval.limit.gmsk.set_iq_offset(value = IqOffsetStruct())
```

Defines and activates upper limits for the I/Q origin offset values.

param value see the help for IqOffsetStruct structure arguments.

set_merror(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk.Gmsk.MerrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:MERRor
driver.configure.multiEval.limit.gmsk.set_merror(value = MerrorStruct())
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the magnitude error.

param value see the help for MerrorStruct structure arguments.

set_perror(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk.Gmsk.PerrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PERRor
driver.configure.multiEval.limit.gmsk.set_perror(value = PerrorStruct())
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the phase error.

param value see the help for PerrorStruct structure arguments.

set_terror(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk.Gmsk.TerrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:TERRor
driver.configure.multiEval.limit.gmsk.set_terror(value = TerrorStruct())
```

Defines and activates upper limits for the timing error.

param value see the help for TerrorStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.clone()
```

Subgroups

7.2.2.8.2.1 PowerVsTime

class PowerVsTime

PowerVsTime commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.clone()
```

Subgroups

7.2.2.8.2.2 Upper

class Upper
Upper commands group definition. 6 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.clone()
```

Subgroups

7.2.2.8.2.3 RisingEdge<RisingEdge>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.repcap_
↪risingEdge_get()
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.repcap_risingEdge_
↪set(repcap.RisingEdge.Nr1)
```

class RisingEdge
RisingEdge commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: RisingEdge, default value after init: RisingEdge.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.clone()
```

Subgroups

7.2.2.8.2.4 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:UPPer:REDGe<RisingEdge>
↪:STATic
```

class Static
Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct
Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s

- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*risingEdge=<RisingEdge.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:REDGe<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.
↪risingEdge.static.get(risingEdge = repcap.RisingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Upper_.RisingEdge_.Static.Static.StaticStruct*, *risingEdge=<RisingEdge.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:REDGe<nr>:STATic
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.static.
↪set(value = [PROPERTY_STRUCT_NAME](), risingEdge = repcap.RisingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

7.2.2.8.2.5 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.dynamic.repcap_
↪rangePcl_set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:UPPer:REDGe<RisingEdge>
↪:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*risingEdge=<RisingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:REDGe<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.
↪risingEdge.dynamic.get(risingEdge = repcap.RisingEdge.Default, rangePcl =␣
↪repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Upper_.RisingEdge_.Dynamic.Dynamic.Dynar*
risingEdge=<RisingEdge.Default: -1>, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:REDGe<nr>:DYNamic<Range>
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.dynamic.
↪set(value = [PROPERTY_STRUCT_NAME](), risingEdge = repcap.RisingEdge.Default,␣
↪rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.risingEdge.dynamic.
↪clone()
```

7.2.2.8.2.6 Upart<UsefulPart>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.repcap_usefulPart_
↪get()
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.repcap_usefulPart_
↪set(repcap.UsefulPart.Nr1)
```

class Upart

Upart commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: UsefulPart, default value after init: UsefulPart.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.clone()
```

Subgroups

7.2.2.8.2.7 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:UPPer:UPARt<UsefulPart>
↪:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*usefulPart=<UsefulPart.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:UPARt<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.
↪upart.static.get(usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Upper_.Upart_.Static.Static.StaticStruct*, *usefulPart=<UsefulPart.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:UPARt<nr>:STATic
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.static.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

7.2.2.8.2.8 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.dynamic.repcap_rangePcl_
↪set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:UPPer:UPARt<UsefulPart>
↪:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*usefulPart*=<*UsefulPart.Default: -1*>, *rangePcl*=<*RangePcl.Default: -1*>) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:UPARt<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.
↪upart.dynamic.get(usefulPart = repcap.UsefulPart.Default, rangePcl = repcap.
↪RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Upper_.Upart_.Dynamic.Dynamic.DynamicSt*
usefulPart=<UsefulPart.Default: -1>, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:UPARt<nr>:DYNamic<Range>
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.dynamic.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default, rangePcl␣
↪= repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.upart.dynamic.clone()
```

7.2.2.8.2.9 FallingEdge<FallingEdge>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.repcap_
↪fallingEdge_get()
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.repcap_fallingEdge_
↪set(repcap.FallingEdge.Nr1)
```

class FallingEdge

FallingEdge commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: FallingEdge, default value after init: FallingEdge.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.clone()
```

Subgroups

7.2.2.8.2.10 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:UPPer:FEDGe
↪<FallingEdge>:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*fallingEdge=<FallingEdge.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:FEDGe<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.
↪fallingEdge.static.get(fallingEdge = repcap.FallingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists

of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Upper_.FallingEdge_.Static.Static.StaticStruct, fallingEdge=<FallingEdge.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:FEDGe<nr>:STATic
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.static.
↪set(value = [PROPERTY_STRUCT_NAME](), fallingEdge = repcap.FallingEdge.
↪Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

7.2.2.8.2.11 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.dynamic.repcap_
↪rangePcl_set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:UPPer:FEDGe
↪<FallingEdge>:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31

- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*fallingEdge=<FallingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:FEDGe<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.
↪fallingEdge.dynamic.get(fallingEdge = repcap.FallingEdge.Default, rangePcl =␣
↪repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Upper_.FallingEdge_.Dynamic.Dynamic.Dyna*
fallingEdge=<FallingEdge.Default: -1>, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:UPPer:FEDGe<nr>:DYNamic<Range>
driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.dynamic.
↪set(value = [PROPERTY_STRUCT_NAME](), fallingEdge = repcap.FallingEdge.
↪Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.upper.fallingEdge.dynamic.
↪clone()
```

7.2.2.8.2.12 Lower

class Lower

Lower commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.lower.clone()
```

Subgroups

7.2.2.8.2.13 Upart<UsefulPart>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.repcap_usefulPart_
↪get()
driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.repcap_usefulPart_
↪set(repcap.UsefulPart.Nr1)
```

class Upart

Upart commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: UsefulPart, default value after init: UsefulPart.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.clone()
```

Subgroups

7.2.2.8.2.14 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:LOWer:UPARt<UsefulPart>
↪:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric End time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric End level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | ON | OFF Start level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start and end level | enables start and end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | ON | OFF End level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start and end level | enables start and end level using the previous/default values)
- Enable: bool: OFF | ON ON: Enable area no OFF: Disable area no

get(*usefulPart=<UsefulPart.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:LOWer:UPARt<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.lower.
↪upart.static.get(usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate lower limit lines for the measured power vs. time. The lines apply to the 'useful part' of a burst for modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line can consist of several areas for which relative and absolute limits can be defined (if both are defined the lower limit overrules the higher one) .

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Lower_.Upart_.Static.Static.StaticStruct*, *usefulPart=<UsefulPart.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:LOWer:UPARt<nr>:STATic
driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.static.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate lower limit lines for the measured power vs. time. The lines apply to the 'useful part' of a burst for modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each

line can consist of several areas for which relative and absolute limits can be defined (if both are defined the lower limit overrules the higher one) .

param structure for set value, see the help for StaticStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

7.2.2.8.2.15 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.dynamic.repcap_rangePcl_
↪set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:PVTime:LOWer:UPARt<UsefulPart>
↪:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*usefulPart=<UsefulPart.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:LOWer:UPARt<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.gmsk.powerVsTime.lower.
↪upart.dynamic.get(usefulPart = repcap.UsefulPart.Default, rangePcl = repcap.
↪RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the lower limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each limit line section can consist of different areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)). In the default configuration, the dynamic corrections for all lower limit lines are set to zero and disabled.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.PowerVsTime_.Lower_.Upart_.Dynamic.Dynamic.DynamicStr usefulPart=<UsefulPart.Default: -1>, rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:PVTime:LOWer:UPARt<nr>:DYNamic<Range>
driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.dynamic.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default, rangePcl␣
↪= repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the lower limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each limit line section can consist of different areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)). In the default configuration, the dynamic corrections for all lower limit lines are set to zero and disabled.

param structure for set value, see the help for DynamicStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.powerVsTime.lower.upart.dynamic.clone()
```

7.2.2.8.2.16 Smodulation

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:SMODulation:RPOWer
```

class Smodulation

Smodulation commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

class RpowerStruct

Structure for reading output parameters. Fields:

- Minimum: float: numeric Low reference power value Range: 0 dBm to 43 dBm, Unit: dBm
- Maximum: float: numeric High reference power value Range: 0 dBm to 43 dBm, Unit: dBm

get_rpower() → RpowerStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SMODulation:RPOWer
value: RpowerStruct = driver.configure.multiEval.limit.gmsk.smodulation.get_
↪rpower()
```

Defines two reference power values for the modulation scheme GMSK. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SMODulation:MPOint<no>.

return structure: for return value, see the help for RpowerStruct structure arguments.

set_rpower(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.Smodulation.Smodulation.RpowerStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SMODulation:RPOWer
driver.configure.multiEval.limit.gmsk.smodulation.set_rpower(value =␣
↪RpowerStruct())
```

Defines two reference power values for the modulation scheme GMSK. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SMODulation:MPOint<no>.

param value see the help for RpowerStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.smodulation.clone()
```

Subgroups

7.2.2.8.2.17 Mpoint<MeasPoint>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.configure.multiEval.limit.gmsk.smodulation.mpoint.repcap_measPoint_get()
driver.configure.multiEval.limit.gmsk.smodulation.mpoint.repcap_measPoint_set(repcap.
↪MeasPoint.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:SMODulation:MPOint<MeasPoint>
```

class Mpoint

Mpoint commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: MeasPoint, default value after init: MeasPoint.Nr1

class MpointStruct

Structure for setting input parameters. Fields:

- Min_Pow_Level_Rel: float: numeric Relative power limit applicable below the low reference power Range: -120 dB to 31.5 dB, Unit: dB
- Max_Pow_Level_Rel: float: numeric Relative power limit applicable above the high reference power Range: -120 dB to 31.5 dB, Unit: dB
- Abs_Power_Level: float: numeric Alternative absolute power limit. If the relative limits are tighter than the absolute limit, the latter applies. Range: -120 dBm to 31.5 dBm, Unit: dBm
- Enable: bool: OFF | ON ON: Enable limits for the given no OFF: Disable limits for the given no

get(*measPoint*=<*MeasPoint.Default: -1*>) → MpointStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SMODulation:MPOint<nr>
value: MpointStruct = driver.configure.multiEval.limit.gmsk.smodulation.mpoint.
↪get(measPoint = repcap.MeasPoint.Default)
```

Defines and activates a limit line for the modulation scheme GMSK for a certain frequency offset. The specified limits apply above the high power reference value and below the low power reference value defined by method RsCmwGsmMeas. Configure.MultiEval.Limit.Gmsk.Smodulation.rpower. Between the two reference power values, the limits are determined by linear interpolation.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

return structure: for return value, see the help for MpointStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.Smodulation_.Mpoint.Mpoint.MpointStruct*, *measPoint*=<*MeasPoint.Default: -1*>) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SMODulation:MPOint<nr>
driver.configure.multiEval.limit.gmsk.smodulation.mpoint.set(value = [PROPERTY_
↪STRUCT_NAME](), measPoint = repcap.MeasPoint.Default)
```

Defines and activates a limit line for the modulation scheme GMSK for a certain frequency offset. The specified limits apply above the high power reference value and below the low power reference value defined by method RsCmwGsmMeas. Configure.MultiEval.Limit.Gmsk.Smodulation.rpower. Between the two reference power values, the limits are determined by linear interpolation.

param structure for set value, see the help for MpointStruct structure arguments.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.smodulation.mpoint.clone()
```

7.2.2.8.2.18 Sswitching

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:SSWitching:PLEVel
```

class Sswitching

Sswitching commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

class PlevelStruct

Structure for reading output parameters. Fields:

- Enable: List[bool]: No parameter help available
- Power_Level: List[float]: No parameter help available

get_plevel() → PlevelStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SSWitching:PLEVel
value: PlevelStruct = driver.configure.multiEval.limit.gmsk.sswitching.get_
↪plevel()
```

Defines and activates reference power values for the modulation scheme GMSK. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SSWitching:MPOint<no>.

return structure: for return value, see the help for PlevelStruct structure arguments.

set_plevel(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.Sswitching.Sswitching.PlevelStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SSWitching:PLEVel
driver.configure.multiEval.limit.gmsk.sswitching.set_plevel(value =␣
↪PlevelStruct())
```

Defines and activates reference power values for the modulation scheme GMSK. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SSWitching:MPOint<no>.

param value see the help for PlevelStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.sswitching.clone()
```

Subgroups

7.2.2.8.2.19 Mpoint<MeasPoint>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.configure.multiEval.limit.gmsk.sswitching.mpoint.repcap_measPoint_get()
driver.configure.multiEval.limit.gmsk.sswitching.mpoint.repcap_measPoint_set(repcap.
↪MeasPoint.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:GMSK:SSWitching:MPOint<MeasPoint>
```

class Mpoint

Mpoint commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: MeasPoint, default value after init: MeasPoint.Nr1

class MpointStruct

Structure for setting input parameters. Fields:

- Limit: List[float]: No parameter help available
- Enable: bool: ON | OFF ON: Enable limits for the given no OFF: Disable limits for the given no

get(*measPoint=<MeasPoint.Default: -1>*) → MpointStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SSWitching:MPOint<nr>
value: MpointStruct = driver.configure.multiEval.limit.gmsk.sswitching.mpoint.
↪get(measPoint = repcap.MeasPoint.Default)
```

Defines and activates a limit line for the modulation scheme GMSK for a certain frequency offset. The specified limits apply at the reference power values defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Gmsk.Sswitching.plevel. Between the reference power values the limits are determined by linear interpolation.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

return structure: for return value, see the help for MpointStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Gmsk_.Sswitching_.Mpoint.Mpoint.MpointStruct*, *measPoint=<MeasPoint.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:GMSK:SSWitching:MPOint<nr>
driver.configure.multiEval.limit.gmsk.sswitching.mpoint.set(value = [PROPERTY_
↪STRUCT_NAME](), measPoint = repcap.MeasPoint.Default)
```

Defines and activates a limit line for the modulation scheme GMSK for a certain frequency offset. The specified limits apply at the reference power values defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Gmsk.Sswitching.plevel. Between the reference power values the limits are determined by linear interpolation.

param structure for set value, see the help for MpointStruct structure arguments.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.gmsk.sswitching.mpoint.clone()
```

7.2.2.8.3 Epsk

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:EVMagnitude
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:MERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:IQOFfset
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:IQIMbalance
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:TERRor
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:FERRor
```

class Epsk

Epsk commands group definition. 19 total commands, 3 Sub-groups, 7 group commands

class EvMagnitudeStruct

Structure for reading output parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

class FreqErrorStruct

Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

class IqImbalanceStruct

Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

class IqOffsetStruct
Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

class MerrorStruct
Structure for reading output parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

class PerrorStruct
Structure for reading output parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

class TerrorStruct
Structure for reading output parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

get_ev_magnitude() → EvMagnitudeStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:EVMagnitude
value: EvMagnitudeStruct = driver.configure.multiEval.limit.epsk.get_ev_
↪magnitude()
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the error vector magnitude (EVM) .

return structure: for return value, see the help for EvMagnitudeStruct structure arguments.

get_freq_error() → FreqErrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:FERRor
value: FreqErrorStruct = driver.configure.multiEval.limit.epsk.get_freq_error()
```

Defines and activates upper limits for the frequency error.

return structure: for return value, see the help for FreqErrorStruct structure arguments.

get_iq_imbalance() → IqImbalanceStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:IQIMbalance
value: IqImbalanceStruct = driver.configure.multiEval.limit.epsk.get_iq_
↪imbalance()
```

Defines and activates upper limits for the I/Q imbalance values.

return structure: for return value, see the help for IqImbalanceStruct structure arguments.

get_iq_offset() → IqOffsetStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:IQOFfset
value: IqOffsetStruct = driver.configure.multiEval.limit.epsk.get_iq_offset()
```

Defines and activates upper limits for the I/Q origin offset values.

return structure: for return value, see the help for IqOffsetStruct structure arguments.

get_merror() → MerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:MERRor
value: MerrorStruct = driver.configure.multiEval.limit.epsk.get_merror()
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the magnitude error.

return structure: for return value, see the help for MerrorStruct structure arguments.

get_perror() → PerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PERRor
value: PerrorStruct = driver.configure.multiEval.limit.epsk.get_perror()
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the phase error.

return structure: for return value, see the help for PerrorStruct structure arguments.

get_terror() → TerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:TERRor
value: TerrorStruct = driver.configure.multiEval.limit.epsk.get_terror()
```

Defines and activates upper limits for the timing error.

return structure: for return value, see the help for TerrorStruct structure arguments.

set_ev_magnitude(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk.Epsk.EvMagnitudeStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:EVMagnitude
driver.configure.multiEval.limit.epsk.set_ev_magnitude(value =␣
↪EvMagnitudeStruct())
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the error vector magnitude (EVM) .

param value see the help for EvMagnitudeStruct structure arguments.

set_freq_error(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk.Epsk.FreqErrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:FERRor
driver.configure.multiEval.limit.epsk.set_freq_error(value = FreqErrorStruct())
```

Defines and activates upper limits for the frequency error.

param value see the help for FreqErrorStruct structure arguments.

set_iq_imbalance(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk.Epsk.IqImbalanceStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:IQIMbalance
driver.configure.multiEval.limit.epsk.set_iq_imbalance(value =␣
↪IqImbalanceStruct())
```

Defines and activates upper limits for the I/Q imbalance values.

param value see the help for IqImbalanceStruct structure arguments.

set_iq_offset(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk.Epsk.IqOffsetStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:IQOFfset
driver.configure.multiEval.limit.epsk.set_iq_offset(value = IqOffsetStruct())
```

Defines and activates upper limits for the I/Q origin offset values.

param value see the help for IqOffsetStruct structure arguments.

set_merror(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk.Epsk.MerrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:MERRor
driver.configure.multiEval.limit.epsk.set_merror(value = MerrorStruct())
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the magnitude error.

param value see the help for MerrorStruct structure arguments.

set_perror(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk.Epsk.PerrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PERRor
driver.configure.multiEval.limit.epsk.set_perror(value = PerrorStruct())
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the phase error.

param value see the help for PerrorStruct structure arguments.

set_terror(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk.Epsk.TerrorStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:TERRor
driver.configure.multiEval.limit.epsk.set_terror(value = TerrorStruct())
```

Defines and activates upper limits for the timing error.

param value see the help for TerrorStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.clone()
```

Subgroups

7.2.2.8.3.1 PowerVsTime

class PowerVsTime

PowerVsTime commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.clone()
```

Subgroups

7.2.2.8.3.2 Upper

class Upper

Upper commands group definition. 6 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.upper.clone()
```

Subgroups

7.2.2.8.3.3 RisingEdge<RisingEdge>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.repcap_
↪risingEdge_get()
driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.repcap_risingEdge_
↪set(repcap.RisingEdge.Nr1)
```

class RisingEdge

RisingEdge commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: RisingEdge, default value after init: RisingEdge.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.clone()
```

Subgroups

7.2.2.8.3.4 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:UPPer:REDGe<RisingEdge>
↪:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)

- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*risingEdge=<RisingEdge.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:REDGe<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.epsk.powerVsTime.upper.
↪risingEdge.static.get(risingEdge = repcap.RisingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Upper_.RisingEdge_.Static.Static.StaticStruct*, *risingEdge=<RisingEdge.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:REDGe<nr>:STATic
driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.static.
↪set(value = [PROPERTY_STRUCT_NAME](), risingEdge = repcap.RisingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

7.2.2.8.3.5 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.dynamic.repcap_
↪rangePcl_set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:UPPer:REDGe<RisingEdge>
↪:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*risingEdge=<RisingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:REDGe<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.epsk.powerVsTime.upper.
↪risingEdge.dynamic.get(risingEdge = repcap.RisingEdge.Default, rangePcl =␣
↪repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Upper_.RisingEdge_.Dynamic.Dynamic.Dynam*
risingEdge=<RisingEdge.Default: -1>, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:REDGe<nr>:DYNamic<Range>
driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.dynamic.
↪set(value = [PROPERTY_STRUCT_NAME](), risingEdge = repcap.RisingEdge.Default,␣
↪rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.upper.risingEdge.dynamic.
↪clone()
```

7.2.2.8.3.6 Upart<UsefulPart>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.repcap_usefulPart_
↪get()
driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.repcap_usefulPart_
↪set(repcap.UsefulPart.Nr1)
```

class Upart

Upart commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: UsefulPart, default value after init: UsefulPart.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.clone()
```

Subgroups

7.2.2.8.3.7 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:UPPer:UPARt<UsefulPart>
↪:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s

- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*usefulPart=<UsefulPart.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:UPARt<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.epsk.powerVsTime.upper.
↪upart.static.get(usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Upper_.Upart_.Static.Static.StaticStruct*, *usefulPart=<UsefulPart.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:UPARt<nr>:STATic
driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.static.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

7.2.2.8.3.8 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.dynamic.repcap_rangePcl_
↪set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:UPPer:UPARt<UsefulPart>
↪:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*usefulPart=<UsefulPart.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:UPARt<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.epsk.powerVsTime.upper.
↪upart.dynamic.get(usefulPart = repcap.UsefulPart.Default, rangePcl = repcap.
↪RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Upper_.Upart_.Dynamic.Dynamic.DynamicStr*
usefulPart=<UsefulPart.Default: -1>, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:UPARt<nr>:DYNamic<Range>
driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.dynamic.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default, rangePcl␣
↪= repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.upper.upart.dynamic.clone()
```

7.2.2.8.3.9 FallingEdge<FallingEdge>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.repcap_
↪fallingEdge_get()
driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.repcap_fallingEdge_
↪set(repcap.FallingEdge.Nr1)
```

class FallingEdge

FallingEdge commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: FallingEdge, default value after init: FallingEdge.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.clone()
```

Subgroups

7.2.2.8.3.10 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:UPPer:FEDGe
↪<FallingEdge>:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*fallingEdge=<FallingEdge.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:FEDGe<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.epsk.powerVsTime.upper.
↪fallingEdge.static.get(fallingEdge = repcap.FallingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Upper_.FallingEdge_.Static.Static.StaticStruct*, *fallingEdge=<FallingEdge.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:FEDGe<nr>:STATic
driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.static.
↪set(value = [PROPERTY_STRUCT_NAME](), fallingEdge = repcap.FallingEdge.
↪Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

7.2.2.8.3.11 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.dynamic.repcap_
↪rangePcl_set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:UPPer:FEDGe
↪<FallingEdge>:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*fallingEdge=<FallingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:FEDGe<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.epsk.powerVsTime.upper.
↪fallingEdge.dynamic.get(fallingEdge = repcap.FallingEdge.Default, rangePcl =␣
↪repcap.RangePcl.Default)
```

(continues on next page)

(continued from previous page)

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Upper_.FallingEdge_.Dynamic.Dynamic.Dynamic*, *fallingEdge=<FallingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:UPPer:FEDGe<nr>:DYNamic<Range>
driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.dynamic.
↪set(value = [PROPERTY_STRUCT_NAME](), fallingEdge = repcap.FallingEdge.
↪Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.upper.fallingEdge.dynamic.
↪clone()
```

7.2.2.8.3.12 Lower

class Lower
Lower commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.lower.clone()
```

Subgroups

7.2.2.8.3.13 Upart<UsefulPart>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.repcap_usefulPart_
↪get()
driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.repcap_usefulPart_
↪set(repcap.UsefulPart.Nr1)
```

class Upart
Upart commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: UsefulPart, default value after init: UsefulPart.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.clone()
```

Subgroups

7.2.2.8.3.14 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:LOWer:UPARt<UsefulPart>
↪:STATic
```

class Static
Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct
Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric End time of the area Range: -50 µs to 600 µs, Unit: s

- Rel_Lev_Start: float: numeric Start level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric End level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | ON | OFF Start level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start and end level | enables start and end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | ON | OFF End level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start and end level | enables start and end level using the previous/default values)
- Enable: bool: OFF | ON ON: Enable area no OFF: Disable area no

get(*usefulPart*=<*UsefulPart.Default: -1*>) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:LOWer:UPARt<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.epsk.powerVsTime.lower.
↪upart.static.get(usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate lower limit lines for the measured power vs. time. The lines apply to the 'useful part' of a burst for modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line can consist of several areas for which relative and absolute limits can be defined (if both are defined the lower limit overrules the higher one) .

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Lower_.Upart_.Static.Static.StaticStruct*, *usefulPart*=<*UsefulPart.Default: -1*>) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:LOWer:UPARt<nr>:STATic
driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.static.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default)
```

These commands define and activate lower limit lines for the measured power vs. time. The lines apply to the 'useful part' of a burst for modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line can consist of several areas for which relative and absolute limits can be defined (if both are defined the lower limit overrules the higher one) .

param structure for set value, see the help for StaticStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

7.2.2.8.3.15 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.dynamic.repcap_rangePcl_
↪set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:PVTime:LOWer:UPARt<UsefulPart>
↪:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*usefulPart=<UsefulPart.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:LOWer:UPARt<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.epsk.powerVsTime.lower.
↪upart.dynamic.get(usefulPart = repcap.UsefulPart.Default, rangePcl = repcap.
↪RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the lower limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each limit line section can consist of different areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)). In the default configuration, the dynamic corrections for all lower limit lines are set to zero and disabled.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.PowerVsTime_.Lower_.Upart_.Dynamic.Dynamic.DynamicStru*
usefulPart=<UsefulPart.Default: -1>, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:PVTime:LOWer:UPARt<nr>:DYNamic<Range>
driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.dynamic.set(value␣
↪= [PROPERTY_STRUCT_NAME](), usefulPart = repcap.UsefulPart.Default, rangePcl␣
↪= repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the lower limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each limit line section can consist of different areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)). In the default configuration, the dynamic corrections for all lower limit lines are set to zero and disabled.

param structure for set value, see the help for DynamicStruct structure arguments.

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.powerVsTime.lower.upart.dynamic.clone()
```

7.2.2.8.3.16 Smodulation

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:SMODulation:RPOWer
```

class Smodulation

Smodulation commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

class RpowerStruct

Structure for reading output parameters. Fields:

- Minimum: float: numeric Low reference power value Range: 0 dBm to 43 dBm, Unit: dBm
- Maximum: float: numeric High reference power value Range: 0 dBm to 43 dBm, Unit: dBm

get_rpower() → RpowerStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SMODulation:RPOWer
value: RpowerStruct = driver.configure.multiEval.limit.epsk.smodulation.get_
↪rpower()
```

Define two reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:EPSK:SMODulation:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SMODulation:MPOint<no>.

return structure: for return value, see the help for RpowerStruct structure arguments.

set_rpower(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.Smodulation.Smodulation.RpowerStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SMODulation:RPOWer
driver.configure.multiEval.limit.epsk.smodulation.set_rpower(value =␣
↪RpowerStruct())
```

Define two reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:EPSK:SMODulation:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SMODulation:MPOint<no>.

param value see the help for RpowerStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.smodulation.clone()
```

Subgroups

7.2.2.8.3.17 Mpoint<MeasPoint>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.configure.multiEval.limit.epsk.smodulation.mpoint.repcap_measPoint_get()
driver.configure.multiEval.limit.epsk.smodulation.mpoint.repcap_measPoint_set(repcap.
↪MeasPoint.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:SMODulation:MPOint<MeasPoint>
```

class Mpoint

Mpoint commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: MeasPoint, default value after init: MeasPoint.Nr1

class MpointStruct

Structure for setting input parameters. Fields:

- Min_Pow_Level_Rel: float: numeric Relative power limit applicable below the low reference power Range: -120 dB to 31.5 dB, Unit: dB
- Max_Pow_Level_Rel: float: numeric Relative power limit applicable above the high reference power Range: -120 dB to 31.5 dB, Unit: dB
- Abs_Power_Level: float: numeric Alternative absolute power limit. If the relative limits are tighter than the absolute limit, the latter applies. Range: -120 dBm to 31.5 dBm, Unit: dBm

- Enable: bool: ON | OFF ON: Enable limits for the given no OFF: Disable limits for the given no

get(*measPoint=<MeasPoint.Default: -1>*) → MpointStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SMODulation:MPOint<nr>
value: MpointStruct = driver.configure.multiEval.limit.epsk.smodulation.mpoint.
↪get(measPoint = repcap.MeasPoint.Default)
```

Defines and activates a limit line for the modulation schemes 8PSK and 16-QAM and for a certain frequency offset. The specified limits apply above the high power reference value and below the low power reference value defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Smodulation.rpower and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam. Smodulation.Rpower.set. Between the two reference power values, the limits are determined by linear interpolation.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

return structure: for return value, see the help for MpointStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.Smodulation_.Mpoint.Mpoint.MpointStruct*, *measPoint=<MeasPoint.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SMODulation:MPOint<nr>
driver.configure.multiEval.limit.epsk.smodulation.mpoint.set(value = [PROPERTY_
↪STRUCT_NAME](), measPoint = repcap.MeasPoint.Default)
```

Defines and activates a limit line for the modulation schemes 8PSK and 16-QAM and for a certain frequency offset. The specified limits apply above the high power reference value and below the low power reference value defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Smodulation.rpower and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam. Smodulation.Rpower.set. Between the two reference power values, the limits are determined by linear interpolation.

param structure for set value, see the help for MpointStruct structure arguments.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.smodulation.mpoint.clone()
```

7.2.2.8.3.18 Sswitching

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:SSWitching:PLEVel
```

class Sswitching

Sswitching commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

class PlevelStruct

Structure for reading output parameters. Fields:

- Enable: List[bool]: No parameter help available
- Power_Level: List[float]: No parameter help available

get_plevel() → PlevelStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SSWitching:PLEVel
value: PlevelStruct = driver.configure.multiEval.limit.epsk.sswitching.get_
↪plevel()
```

Define and activate reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SSWitching:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SSWitching:MPOint<no>.

return structure: for return value, see the help for PlevelStruct structure arguments.

set_plevel(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.Sswitching.Sswitching.PlevelStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SSWitching:PLEVel
driver.configure.multiEval.limit.epsk.sswitching.set_plevel(value =␣
↪PlevelStruct())
```

Define and activate reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SSWitching:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SSWitching:MPOint<no>.

param value see the help for PlevelStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.sswitching.clone()
```

Subgroups

7.2.2.8.3.19 Mpoint<MeasPoint>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.configure.multiEval.limit.epsk.sswitching.mpoint.repcap_measPoint_get()
driver.configure.multiEval.limit.epsk.sswitching.mpoint.repcap_measPoint_set(repcap.
↪MeasPoint.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:EPSK:SSWitching:MPOint<MeasPoint>
```

class Mpoint

Mpoint commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: MeasPoint, default value after init: MeasPoint.Nr1

class MpointStruct

Structure for setting input parameters. Fields:

- Limit: List[float]: No parameter help available
- Enable: bool: OFF | ON ON: Enable limits for the given no OFF: Disable limits for the given no

get(*measPoint=<MeasPoint.Default: -1>*) → MpointStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SSWitching:MPOint<nr>
value: MpointStruct = driver.configure.multiEval.limit.epsk.sswitching.mpoint.
↪get(measPoint = repcap.MeasPoint.Default)
```

Define and activate a limit line for the modulation schemes 8PSK and 16-QAM for a certain frequency offset. The specified limits apply at the reference power values defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Sswitching. plevel and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam.Sswitching.Plevel.set. Between the reference power values the limits are determined by linear interpolation.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

return structure: for return value, see the help for MpointStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Epsk_.Sswitching_.Mpoint.Mpoint.MpointStruct*, *measPoint=<MeasPoint.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>
↪:MEValuation:LIMit:EPSK:SSWitching:MPOint<nr>
driver.configure.multiEval.limit.epsk.sswitching.mpoint.set(value = [PROPERTY_
↪STRUCT_NAME](), measPoint = repcap.MeasPoint.Default)
```

Define and activate a limit line for the modulation schemes 8PSK and 16-QAM for a certain frequency offset. The specified limits apply at the reference power values defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Sswitching. plevel and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam.Sswitching.Plevel.set. Between the reference power values the limits are determined by linear interpolation.

param structure for set value, see the help for MpointStruct structure arguments.

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.epsk.sswitching.mpoint.clone()
```

7.2.2.8.4 Qam<QamOrder>

RepCap Settings

```
# Range: Nr16 .. Nr16
rc = driver.configure.multiEval.limit.qam.repcap_qamOrder_get()
driver.configure.multiEval.limit.qam.repcap_qamOrder_set(repcap.QamOrder.Nr16)
```

class Qam

Qam commands group definition. 19 total commands, 10 Sub-groups, 0 group commands Repeated Capability: QamOrder, default value after init: QamOrder.Nr16

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.clone()
```

Subgroups

7.2.2.8.4.1 PowerVsTime

class PowerVsTime

PowerVsTime commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.clone()
```

Subgroups

7.2.2.8.4.2 Upper

class Upper

Upper commands group definition. 6 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.upper.clone()
```

Subgroups

7.2.2.8.4.3 RisingEdge<RisingEdge>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.repcap_risingEdge_
↪get()
driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.repcap_risingEdge_
↪set(repcap.RisingEdge.Nr1)
```

class RisingEdge

RisingEdge commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: RisingEdge, default value after init: RisingEdge.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.clone()
```

Subgroups

7.2.2.8.4.4 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:UPPer:REDGe
↪<RisingEdge>:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*qamOrder*=<*QamOrder.Default: -1*>, *risingEdge*=<*RisingEdge.Default: -1*>) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:REDGe<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.qam.powerVsTime.upper.
↪risingEdge.static.get(qamOrder = repcap.QamOrder.Default, risingEdge = repcap.
↪RisingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Upper_.RisingEdge_.Static.Static.StaticStruct*, *qamOrder=<QamOrder.Default: -1>*, *risingEdge=<RisingEdge.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:REDGe<nr>:STATic
driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.static.
↪set(value = [PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default,␣
↪risingEdge = repcap.RisingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

7.2.2.8.4.5 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.dynamic.repcap_
↪rangePcl_set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:UPPer:REDGe
↪<RisingEdge>:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*qamOrder=<QamOrder.Default: -1>*, *risingEdge=<RisingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:REDGe<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.qam.powerVsTime.upper.
↪risingEdge.dynamic.get(qamOrder = repcap.QamOrder.Default, risingEdge =␣
↪repcap.RisingEdge.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Upper_.RisingEdge_.Dynamic.Dynamic.Dynam*
qamOrder=<QamOrder.Default: -1>, *risingEdge=<RisingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:REDGe<nr>:DYNamic<Range>
driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.dynamic.
↪set(value = [PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default,␣
↪risingEdge = repcap.RisingEdge.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param risingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'RisingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.upper.risingEdge.dynamic.
↪clone()
```

7.2.2.8.4.6 Upart<UsefulPart>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.repcap_usefulPart_get()
driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.repcap_usefulPart_
↪set(repcap.UsefulPart.Nr1)
```

class Upart

Upart commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: UsefulPart, default value after init: UsefulPart.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.clone()
```

Subgroups

7.2.2.8.4.7 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:UPPer:UPARt
↪<UsefulPart>:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)

- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*qamOrder=<QamOrder.Default: -1>*, *usefulPart=<UsefulPart.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:UPARt<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.qam.powerVsTime.upper.
↪upart.static.get(qamOrder = repcap.QamOrder.Default, usefulPart = repcap.
↪UsefulPart.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Upper_.Upart_.Static.Static.StaticStruct*, *qamOrder=<QamOrder.Default: -1>*, *usefulPart=<UsefulPart.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:UPARt<nr>:STATic
driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.static.set(value =␣
↪[PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default, usefulPart =␣
↪repcap.UsefulPart.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

7.2.2.8.4.8 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.dynamic.repcap_rangePcl_
↪set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:UPPer:UPARt
↪<UsefulPart>:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*qamOrder=<QamOrder.Default: -1>*, *usefulPart=<UsefulPart.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:UPARt<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.qam.powerVsTime.upper.
↪upart.dynamic.get(qamOrder = repcap.QamOrder.Default, usefulPart = repcap.
↪UsefulPart.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Upper_.Upart_.Dynamic.Dynamic.DynamicStr*
qamOrder=<QamOrder.Default: -1>, *usefulPart=<UsefulPart.Default: -1>*,
rangePcl=<RangePcl.Default: -1>) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:UPARt<nr>:DYNamic<Range>
driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.dynamic.set(value␣
↪= [PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default, usefulPart =␣
↪repcap.UsefulPart.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.upper.upart.dynamic.clone()
```

7.2.2.8.4.9 FallingEdge<FallingEdge>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.repcap_
↪fallingEdge_get()
driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.repcap_fallingEdge_
↪set(repcap.FallingEdge.Nr1)
```

class FallingEdge

FallingEdge commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: FallingEdge, default value after init: FallingEdge.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.clone()
```

Subgroups

7.2.2.8.4.10 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:UPPer:FEDGe
↪<FallingEdge>:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric Start and end time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric Start and end level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | OFF | ON Start and end level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start/end level | enables start/end level using the previous/default values)
- Enable: bool: ON | OFF ON: Enable area no OFF: Disable area no

get(*qamOrder=<QamOrder.Default: -1>*, *fallingEdge=<FallingEdge.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:FEDGe<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.qam.powerVsTime.upper.
↪fallingEdge.static.get(qamOrder = repcap.QamOrder.Default, fallingEdge =␣
↪repcap.FallingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Upper_.FallingEdge_.Static.Static.StaticStruct, qamOrder=<QamOrder.Default: -1>, fallingEdge=<FallingEdge.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:FEDGe<nr>:STATic
driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.static.
↪set(value = [PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default,␣
↪fallingEdge = repcap.FallingEdge.Default)
```

These commands define and activate upper limit lines for the measured power vs. time. The lines apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line consists of three sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each section consists of several areas for which relative and absolute limits can be defined (if both are defined the higher limit overrules the lower one) .

param structure for set value, see the help for StaticStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

7.2.2.8.4.11 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.dynamic.repcap_
↪rangePcl_set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:UPPer:FEDGe
↪<FallingEdge>:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31

- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*qamOrder=<QamOrder.Default: -1>*, *fallingEdge=<FallingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:FEDGe<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.qam.powerVsTime.upper.
↪fallingEdge.dynamic.get(qamOrder = repcap.QamOrder.Default, fallingEdge =␣
↪repcap.FallingEdge.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Upper_.FallingEdge_.Dynamic.Dynamic.Dynar*, *qamOrder=<QamOrder.Default: -1>*, *fallingEdge=<FallingEdge.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:UPPer:FEDGe<nr>:DYNamic<Range>
driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.dynamic.
↪set(value = [PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default,␣
↪fallingEdge = repcap.FallingEdge.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the upper limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) and to the three limit line sections: rising edge (REDGe) , useful part (UPARt) and falling edge (FEDGe) . Each limit line section consists of several areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)).

param structure for set value, see the help for DynamicStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param fallingEdge optional repeated capability selector. Default value: Nr1 (settable in the interface 'FallingEdge')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.upper.fallingEdge.dynamic.
↪clone()
```

7.2.2.8.4.12 Lower

class Lower

Lower commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.lower.clone()
```

Subgroups

7.2.2.8.4.13 Upart<UsefulPart>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.repcap_usefulPart_get()
driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.repcap_usefulPart_
↪set(repcap.UsefulPart.Nr1)
```

class Upart

Upart commands group definition. 2 total commands, 2 Sub-groups, 0 group commands Repeated Capability: UsefulPart, default value after init: UsefulPart.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.clone()
```

Subgroups

7.2.2.8.4.14 Static

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:LOWer:UPARt
↪<UsefulPart>:STATic
```

class Static

Static commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class StaticStruct

Structure for setting input parameters. Fields:

- Time_Start: float: numeric Start time of the area Range: -50 µs to 600 µs, Unit: s
- Time_End: float: numeric End time of the area Range: -50 µs to 600 µs, Unit: s
- Rel_Lev_Start: float: numeric Start level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Rel_Lev_End: float: numeric End level of the relative limit for the area Range: -100 dB to 10 dB, Unit: dB
- Abs_Lev_Start: float or bool: numeric | ON | OFF Start level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start and end level | enables start and end level using the previous/default values)
- Abs_Lev_End: float or bool: numeric | ON | OFF End level of the absolute limit for the area Range: -100 dBm to 10 dBm, Unit: dBm Additional parameters: OFF | ON (disables start and end level | enables start and end level using the previous/default values)
- Enable: bool: OFF | ON ON: Enable area no OFF: Disable area no

get(*qamOrder=<QamOrder.Default: -1>*, *usefulPart=<UsefulPart.Default: -1>*) → StaticStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:LOWer:UPARt<nr>:STATic
value: StaticStruct = driver.configure.multiEval.limit.qam.powerVsTime.lower.
↪upart.static.get(qamOrder = repcap.QamOrder.Default, usefulPart = repcap.
↪UsefulPart.Default)
```

These commands define and activate lower limit lines for the measured power vs. time. The lines apply to the 'useful part' of a burst for modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line can consist of several areas for which relative and absolute limits can be defined (if both are defined the lower limit overrules the higher one) .

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

return structure: for return value, see the help for StaticStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Lower_.Upart_.Static.Static.StaticStruct*, *qamOrder=<QamOrder.Default: -1>*, *usefulPart=<UsefulPart.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:LOWer:UPARt<nr>:STATic
driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.static.set(value =␣
↪[PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default, usefulPart =␣
↪repcap.UsefulPart.Default)
```

These commands define and activate lower limit lines for the measured power vs. time. The lines apply to the 'useful part' of a burst for modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each line can consist of several areas for which relative and absolute limits can be defined (if both are defined the lower limit overrules the higher one) .

param structure for set value, see the help for StaticStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

7.2.2.8.4.15 Dynamic<RangePcl>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.dynamic.repcap_
↪rangePcl_get()
driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.dynamic.repcap_rangePcl_
↪set(repcap.RangePcl.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PVTime:LOWer:UPARt
↪<UsefulPart>:DYNamic<RangePcl>
```

class Dynamic

Dynamic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: RangePcl, default value after init: RangePcl.Nr1

class DynamicStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF | ON Disable or enable dynamic correction
- Pcl_Start: float: numeric First PCL in PCL range Range: 0 to 31
- Pcl_End: float: numeric Last PCL in PCL range (can be equal to PCLStart) Range: 0 to 31
- Correction: float: numeric Correction value for power template Range: -100 dB to 100 dB, Unit: dB

get(*qamOrder=<QamOrder.Default: -1>*, *usefulPart=<UsefulPart.Default: -1>*, *rangePcl=<RangePcl.Default: -1>*) → DynamicStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:LOWer:UPARt<nr>:DYNamic<Range>
value: DynamicStruct = driver.configure.multiEval.limit.qam.powerVsTime.lower.
↪upart.dynamic.get(qamOrder = repcap.QamOrder.Default, usefulPart = repcap.
↪UsefulPart.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the lower limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each limit line section can consist of different areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)). In the default configuration, the dynamic corrections for all lower limit lines are set to zero and disabled.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

return structure: for return value, see the help for DynamicStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.PowerVsTime_.Lower_.Upart_.Dynamic.Dynamic.DynamicStru*
qamOrder=<QamOrder.Default: -1>, usefulPart=<UsefulPart.Default: -1>,
rangePcl=<RangePcl.Default: -1>) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PVTime:LOWer:UPARt<nr>:DYNamic<Range>
driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.dynamic.set(value␣
↪= [PROPERTY_STRUCT_NAME](), qamOrder = repcap.QamOrder.Default, usefulPart =␣
↪repcap.UsefulPart.Default, rangePcl = repcap.RangePcl.Default)
```

These commands define and activate dynamic (PCL-dependent) corrections to the lower limit lines for the measured power vs. time. The corrections apply to the modulation schemes GMSK, 8PSK (EPSK) or 16-QAM (QAM16) . Each limit line section can consist of different areas (<no>) . Each dynamic correction is defined for up to five different PCL ranges (<Range>)). In the default configuration, the dynamic corrections for all lower limit lines are set to zero and disabled.

param structure for set value, see the help for DynamicStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param usefulPart optional repeated capability selector. Default value: Nr1 (settable in the interface 'Upart')

param rangePcl optional repeated capability selector. Default value: Nr1 (settable in the interface 'Dynamic')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.powerVsTime.lower.upart.dynamic.clone()
```

7.2.2.8.4.16 EvMagnitude

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:EVMagnitude
```

class EvMagnitude

EvMagnitude commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class EvMagnitudeStruct

Structure for setting input parameters. Fields:

- Values: List[float]: No parameter help available

- Selection: List[bool]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → EvMagnitudeStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:EVMagnitude
value: EvMagnitudeStruct = driver.configure.multiEval.limit.qam.evMagnitude.
↪get(qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the error vector magnitude (EVM) .

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for EvMagnitudeStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.EvMagnitude.EvMagnitude.EvMagnitudeStruct*, *qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:EVMagnitude
driver.configure.multiEval.limit.qam.evMagnitude.set(value = [PROPERTY_STRUCT_
↪NAME](), qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the error vector magnitude (EVM) .

param structure for set value, see the help for EvMagnitudeStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.17 Merror

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:MERRor
```

class Merror

Merror commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class MerrorStruct

Structure for setting input parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → MerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:MERRor
value: MerrorStruct = driver.configure.multiEval.limit.qam.merror.get(qamOrder␣
↪= repcap.QamOrder.Default)
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the magnitude error.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for MerrorStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.Merror.Merror.MerrorStruct*, *qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:MERRor
driver.configure.multiEval.limit.qam.merror.set(value = [PROPERTY_STRUCT_
↪NAME](), qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the magnitude error.

param structure for set value, see the help for MerrorStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.18 Perror

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:PERRor
```

class Perror

Perror commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class PerrorStruct

Structure for setting input parameters. Fields:

- Values: List[float]: No parameter help available
- Selection: List[bool]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → PerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PERRor
value: PerrorStruct = driver.configure.multiEval.limit.qam.perror.get(qamOrder␣
↪= repcap.QamOrder.Default)
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the phase error.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for PerrorStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.Perror.Perror.PerrorStruct, qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PERRor
driver.configure.multiEval.limit.qam.perror.set(value = [PROPERTY_STRUCT_
↪NAME](), qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the RMS, peak and 95th percentile values of the phase error.

param structure for set value, see the help for PerrorStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.19 IqOffset

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:IQOFfset
```

class IqOffset

IqOffset commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class IqOffsetStruct

Structure for setting input parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → IqOffsetStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQOFfset
value: IqOffsetStruct = driver.configure.multiEval.limit.qam.iqOffset.
↪get(qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the I/Q origin offset values.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for IqOffsetStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.IqOffset.IqOffset.IqOffsetStruct, qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQOFfset
driver.configure.multiEval.limit.qam.iqOffset.set(value = [PROPERTY_STRUCT_
↪NAME](), qamOrder = repcap.QamOrder.Default)
```

(continues on next page)

(continued from previous page)

Defines and activates upper limits for the I/Q origin offset values.

param structure for set value, see the help for IqOffsetStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.20 IqImbalance

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:IQIMbalance
```

class IqImbalance

IqImbalance commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class IqImbalanceStruct

Structure for setting input parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → IqImbalanceStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQIMbalance
value: IqImbalanceStruct = driver.configure.multiEval.limit.qam.iqImbalance.
↪get(qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the I/Q imbalance values.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for IqImbalanceStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.IqImbalance.IqImbalance.IqImbalanceStruct*, *qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQIMbalance
driver.configure.multiEval.limit.qam.iqImbalance.set(value = [PROPERTY_STRUCT_
↪NAME](), qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the I/Q imbalance values.

param structure for set value, see the help for IqImbalanceStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.21 Terror

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:TERRor
```

class Terror

Terror commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class TerrorStruct

Structure for setting input parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → TerrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:TERRor
value: TerrorStruct = driver.configure.multiEval.limit.qam.terror.get(qamOrder␣
↪= repcap.QamOrder.Default)
```

Defines and activates upper limits for the timing error.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for TerrorStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.Terror.Terror.TerrorStruct*, *qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:TERRor
driver.configure.multiEval.limit.qam.terror.set(value = [PROPERTY_STRUCT_
↪NAME](), qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the timing error.

param structure for set value, see the help for TerrorStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.22 FreqError

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:FERRor
```

class FreqError

FreqError commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FreqErrorStruct

Structure for setting input parameters. Fields:

- Value: float: No parameter help available
- Selection: List[bool]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → FreqErrorStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:FERRor
value: FreqErrorStruct = driver.configure.multiEval.limit.qam.freqError.
↪get(qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the frequency error.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for FreqErrorStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.FreqError.FreqError.FreqErrorStruct*, *qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:FERRor
driver.configure.multiEval.limit.qam.freqError.set(value = [PROPERTY_STRUCT_
↪NAME](), qamOrder = repcap.QamOrder.Default)
```

Defines and activates upper limits for the frequency error.

param structure for set value, see the help for FreqErrorStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.23 Smodulation

class Smodulation

Smodulation commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.smodulation.clone()
```

Subgroups

7.2.2.8.4.24 Rpower

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:SMODulation:RPOWer
```

class Rpower

Rpower commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class RpowerStruct

Structure for setting input parameters. Fields:

- Minimum: float: numeric Low reference power value Range: 0 dBm to 43 dBm, Unit: dBm
- Maximum: float: numeric High reference power value Range: 0 dBm to 43 dBm, Unit: dBm

get(*qamOrder=<QamOrder.Default: -1>*) → RpowerStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SMODulation:RPOWer
value: RpowerStruct = driver.configure.multiEval.limit.qam.smodulation.rpower.
↪get(qamOrder = repcap.QamOrder.Default)
```

Define two reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:EPSK:SMODulation:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SMODulation:MPOint<no>.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for RpowerStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.Smodulation_.Rpower.Rpower.RpowerStruct*, *qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SMODulation:RPOWer
driver.configure.multiEval.limit.qam.smodulation.rpower.set(value = [PROPERTY_
↪STRUCT_NAME](), qamOrder = repcap.QamOrder.Default)
```

(continues on next page)

(continued from previous page)

Define two reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:EPSK:SMODulation:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SMODulation:MPOint<no>.

param structure for set value, see the help for RpowerStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.25 Mpoint<MeasPoint>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.configure.multiEval.limit.qam.smodulation.mpoint.repcap_measPoint_get()
driver.configure.multiEval.limit.qam.smodulation.mpoint.repcap_measPoint_set(repcap.
↪MeasPoint.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:SMODulation:MPOint
↪<MeasPoint>
```

class Mpoint

Mpoint commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: MeasPoint, default value after init: MeasPoint.Nr1

class MpointStruct

Structure for setting input parameters. Fields:

- Min_Pow_Level_Rel: float: numeric Relative power limit applicable below the low reference power Range: -120 dB to 31.5 dB, Unit: dB
- Max_Pow_Level_Rel: float: numeric Relative power limit applicable above the high reference power Range: -120 dB to 31.5 dB, Unit: dB
- Abs_Power_Level: float: numeric Alternative absolute power limit. If the relative limits are tighter than the absolute limit, the latter applies. Range: -120 dBm to 31.5 dBm, Unit: dBm
- Enable: bool: ON | OFF ON: Enable limits for the given no OFF: Disable limits for the given no

get(*qamOrder=<QamOrder.Default: -1>*, *measPoint=<MeasPoint.Default: -1>*) → MpointStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SMODulation:MPOint<nr>
value: MpointStruct = driver.configure.multiEval.limit.qam.smodulation.mpoint.
↪get(qamOrder = repcap.QamOrder.Default, measPoint = repcap.MeasPoint.Default)
```

Defines and activates a limit line for the modulation schemes 8PSK and 16-QAM and for a certain frequency offset. The specified limits apply above the high power reference value and below the low power reference value defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Smodulation.rpower

and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam. Smodulation.Rpower.set. Between the two reference power values, the limits are determined by linear interpolation.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

return structure: for return value, see the help for MpointStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.Smodulation_.Mpoint.Mpoint.MpointStruct*, *qamOrder=<QamOrder.Default: -1>*, *measPoint=<MeasPoint.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SMODulation:MPOint<nr>
driver.configure.multiEval.limit.qam.smodulation.mpoint.set(value = [PROPERTY_
↪STRUCT_NAME](), qamOrder = repcap.QamOrder.Default, measPoint = repcap.
↪MeasPoint.Default)
```

Defines and activates a limit line for the modulation schemes 8PSK and 16-QAM and for a certain frequency offset. The specified limits apply above the high power reference value and below the low power reference value defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Smodulation.rpower and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam. Smodulation.Rpower.set. Between the two reference power values, the limits are determined by linear interpolation.

param structure for set value, see the help for MpointStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.smodulation.mpoint.clone()
```

7.2.2.8.4.26 Sswitching

class Sswitching

Sswitching commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.sswitching.clone()
```

Subgroups

7.2.2.8.4.27 Plevel

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:SSWitching:PLEVel
```

class Plevel

Plevel commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class PlevelStruct

Structure for setting input parameters. Fields:

- Enable: List[bool]: No parameter help available
- Power_Level: List[float]: No parameter help available

get(*qamOrder=<QamOrder.Default: -1>*) → PlevelStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SSWitching:PLEVel
value: PlevelStruct = driver.configure.multiEval.limit.qam.sswitching.plevel.
↪get(qamOrder = repcap.QamOrder.Default)
```

Define and activate reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SSWitching:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SSWitching:MPOint<no>.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

return structure: for return value, see the help for PlevelStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.Sswitching_.Plevel.Plevel.PlevelStruct*, *qamOrder=<QamOrder.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SSWitching:PLEVel
driver.configure.multiEval.limit.qam.sswitching.plevel.set(value = [PROPERTY_
↪STRUCT_NAME](), qamOrder = repcap.QamOrder.Default)
```

Define and activate reference power values for the modulation schemes 8PSK and 16-QAM. These values are relevant in the context of CONFigure:GSM:MEAS<i>:MEValuation:LIMit:GMSK:SSWitching:MPOint<no> and CONFigure:GSM:MEAS<i>:MEValuation:LIMit:QAM<m>:SSWitching:MPOint<no>.

param structure for set value, see the help for PlevelStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

7.2.2.8.4.28 Mpoint<MeasPoint>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.configure.multiEval.limit.qam.sswitching.mpoint.repcap_measPoint_get()
driver.configure.multiEval.limit.qam.sswitching.mpoint.repcap_measPoint_set(repcap.
↪MeasPoint.Nr1)
```

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<QamOrder>:SSWitching:MPOint
↪<MeasPoint>
```

class Mpoint

Mpoint commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: MeasPoint, default value after init: MeasPoint.Nr1

class MpointStruct

Structure for setting input parameters. Fields:

- Limit: List[float]: No parameter help available
- Enable: bool: OFF | ON ON: Enable limits for the given no OFF: Disable limits for the given no

get(*qamOrder=<QamOrder.Default: -1>*, *measPoint=<MeasPoint.Default: -1>*) → MpointStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SSWitching:MPOint<nr>
value: MpointStruct = driver.configure.multiEval.limit.qam.sswitching.mpoint.
↪get(qamOrder = repcap.QamOrder.Default, measPoint = repcap.MeasPoint.Default)
```

Define and activate a limit line for the modulation schemes 8PSK and 16-QAM for a certain frequency offset. The specified limits apply at the reference power values defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Sswitching. plevel and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam.Sswitching.Plevel.set. Between the reference power values the limits are determined by linear interpolation.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

return structure: for return value, see the help for MpointStruct structure arguments.

set(*structure: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Limit_.Qam_.Sswitching_.Mpoint.Mpoint.MpointStruct*, *qamOrder=<QamOrder.Default: -1>*, *measPoint=<MeasPoint.Default: -1>*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SSWitching:MPOint<nr>
driver.configure.multiEval.limit.qam.sswitching.mpoint.set(value = [PROPERTY_
↪STRUCT_NAME](), qamOrder = repcap.QamOrder.Default, measPoint = repcap.
↪MeasPoint.Default)
```

Define and activate a limit line for the modulation schemes 8PSK and 16-QAM for a certain frequency offset. The specified limits apply at the reference power values defined by method RsCmwGsmMeas.Configure.MultiEval.Limit.Epsk.Sswitching. plevel and method RsCmwGsmMeas.Configure.MultiEval.Limit.Qam.Sswitching.Plevel.set. Between the reference power values the limits are determined by linear interpolation.

param structure for set value, see the help for MpointStruct structure arguments.

param qamOrder optional repeated capability selector. Default value: Nr16 (settable in the interface 'Qam')

param measPoint optional repeated capability selector. Default value: Nr1 (settable in the interface 'Mpoint')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.multiEval.limit.qam.sswitching.mpoint.clone()
```

7.2.2.9 Smodulation

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:OFRequence
CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:EAREa
CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:TDFSelect
```

class Smodulation

Smodulation commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class EareaStruct

Structure for reading output parameters. Fields:

- Enable_1: bool: OFF | ON ON: Enable area 1 OFF: Disable area 1
- Start_1: int: integer Start of evaluation area 1 Range: 0 Sym to 146 Sym, Unit: Symbol
- Stop_1: int: integer Stop of evaluation area 1 Range: 1 Symbol to 147 Symbol, Unit: Symbol
- Enable_2: bool: OFF | ON ON: Enable area 2 OFF: Disable area 2
- Start_2: int: integer Start of evaluation area 2 Range: 0 Sym to 146 Sym, Unit: Symbol
- Stop_2: int: integer Stop of evaluation area 2 Range: 1 Symbol to 147 Symbol, Unit: Symbol

get_earea() → EareaStruct

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:EAREa
value: EareaStruct = driver.configure.multiEval.smodulation.get_earea()
```

Defines the time intervals (evaluation areas) to be used for spectrum modulation measurements.

return structure: for return value, see the help for EareaStruct structure arguments.

get_ofrequence() → List[float]

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:OFRequence
value: List[float or bool] = driver.configure.multiEval.smodulation.get_
↪ofrequence()
```

Defines the frequency offsets to be used for spectrum modulation measurements. The offsets are defined relative to the analyzer frequency. Up to 20 offsets can be defined and enabled.

return frequency_offset: No help available

get_tdf_select() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:TDFSelect
value: int or bool = driver.configure.multiEval.smodulation.get_tdf_select()
```

Defines the offset frequency for the spectrum modulation time diagram. The diagram shows the measured power vs. time at the selected offset frequency. The numbers 1 to 20 select the negative frequency offsets from the frequency offsets list, numbers 21 to 40 select the positive frequency offsets.

return nr_freq_offset: integer | ON | OFF Range: 0 to 40 Additional parameters: ON | OFF (enables | disables offset)

set_earea(*value: RsCmwGsmMeas.Implementations.Configure_.MultiEval_.Smodulation.Smodulation.EareaStruct*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:EARea
driver.configure.multiEval.smodulation.set_earea(value = EareaStruct())
```

Defines the time intervals (evaluation areas) to be used for spectrum modulation measurements.

param value see the help for EareaStruct structure arguments.

set_ofrequence(*frequency_offset: List[float]*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:OFRequence
driver.configure.multiEval.smodulation.set_ofrequence(frequency_offset = [1.1,␣
↪True, 2.2, False, 3.3])
```

Defines the frequency offsets to be used for spectrum modulation measurements. The offsets are defined relative to the analyzer frequency. Up to 20 offsets can be defined and enabled.

param frequency_offset numeric | OFF | ON Set and enable frequency offset. Range: 0 Hz to 3 MHz, Unit: Hz Additional parameters: OFF | ON (disables / enables offset using the previous/default value)

set_tdf_select(*nr_freq_offset: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SMODulation:TDFSelect
driver.configure.multiEval.smodulation.set_tdf_select(nr_freq_offset = 1)
```

Defines the offset frequency for the spectrum modulation time diagram. The diagram shows the measured power vs. time at the selected offset frequency. The numbers 1 to 20 select the negative frequency offsets from the frequency offsets list, numbers 21 to 40 select the positive frequency offsets.

param nr_freq_offset integer | ON | OFF Range: 0 to 40 Additional parameters: ON | OFF (enables | disables offset)

7.2.2.10 Sswitching

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:OFRequence
CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:TDFSelect
CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:PHMode
```

class Sswitching

Sswitching commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_ofrequence() → List[float]

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:OFRequence
value: List[float or bool] = driver.configure.multiEval.sswitching.get_
↪ofrequence()
```

Defines the frequency offsets to be used for spectrum switching measurements. The offsets are defined relative to the analyzer frequency. Up to 20 offsets can be defined and enabled.

return frequency_offset: No help available

get_ph_mode() → RsCmwGsmMeas.enums.PeakHoldMode

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:PHMode
value: enums.PeakHoldMode = driver.configure.multiEval.sswitching.get_ph_mode()
```

Specifies how the peak hold mode is used for the spectrum switching results in frequency domain (bar graphs) and in time domain.

return peak_hold_mode: PHOL | SCO PHOL: Frequency and time: peak hold SCO: Frequency: stat. count, time: current

get_tdf_select() → int

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:TDFSelect
value: int or bool = driver.configure.multiEval.sswitching.get_tdf_select()
```

Defines the offset frequency for the spectrum modulation time diagram. The diagram shows the measured power vs. time at the selected offset frequency. The numbers 1 to 20 select the negative frequency offsets from the frequency offsets list, numbers 21 to 40 select the positive frequency offsets.

return nr_freq_offset: integer | ON | OFF Range: 0 to 40 Additional parameters: OFF | ON (disables | enables the offset)

set_ofrequence(*frequency_offset: List[float]*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:OFRequence
driver.configure.multiEval.sswitching.set_ofrequence(frequency_offset = [1.1,␣
↪True, 2.2, False, 3.3])
```

Defines the frequency offsets to be used for spectrum switching measurements. The offsets are defined relative to the analyzer frequency. Up to 20 offsets can be defined and enabled.

param frequency_offset numeric | OFF | ON Set and enable frequency offset. Range: 0 Hz to 3 MHz, Unit: Hz Additional parameters: OFF | ON (disables / enables offset using the previous/default value)

set_ph_mode(*peak_hold_mode: RsCmwGsmMeas.enums.PeakHoldMode*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:PHMode
driver.configure.multiEval.sswitching.set_ph_mode(peak_hold_mode = enums.
↪PeakHoldMode.PHOL)
```

Specifies how the peak hold mode is used for the spectrum switching results in frequency domain (bar graphs) and in time domain.

param peak_hold_mode PHOL | SCO PHOL: Frequency and time: peak hold SCO: Frequency: stat. count, time: current

set_tdf_select(*nr_freq_offset: int*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:SSWitching:TDFSelect
driver.configure.multiEval.sswitching.set_tdf_select(nr_freq_offset = 1)
```

Defines the offset frequency for the spectrum modulation time diagram. The diagram shows the measured power vs. time at the selected offset frequency. The numbers 1 to 20 select the negative frequency offsets from the frequency offsets list, numbers 21 to 40 select the positive frequency offsets.

param nr_freq_offset integer | ON | OFF Range: 0 to 40 Additional parameters: OFF | ON (disables | enables the offset)

7.2.2.11 Ber

SCPI Commands

```
CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:LOOP
CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:TSTart
CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:TRUN
```

class Ber

Ber commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_loop() → RsCmwGsmMeas.enums.LoopType

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:LOOP
value: enums.LoopType = driver.configure.multiEval.ber.get_loop()
```

Selects the loop for BER tests.

return loop: C | SRB C: Loop C (for GMSK signals, with channel coding) SRB: SRB loop (for 8PSK-modulated signals, MCS7 to MCS9)

get_trun() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:TRUN
value: float = driver.configure.multiEval.ber.get_trun()
```

Selects the threshold run value for BER tests. This value is the maximum bit error rate in any burst considered for the BER measurement.

return threshold_run: numeric Range: 0 % to 100 %, Unit: %

get_tstart() → float

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:TSTart
value: float = driver.configure.multiEval.ber.get_tstart()
```

Selects the threshold start value for BER tests. This value is the maximum bit error rate in the first burst of the BER measurement.

return threshold_start: numeric Range: 0 % to 100 %, Unit: %

set_loop(*loop: RsCmwGsmMeas.enums.LoopType*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:LOOP
driver.configure.multiEval.ber.set_loop(loop = enums.LoopType.C)
```

Selects the loop for BER tests.

param loop C | SRB C: Loop C (for GMSK signals, with channel coding) SRB: SRB loop (for 8PSK-modulated signals, MCS7 to MCS9)

set_trun(*threshold_run: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:TRUN
driver.configure.multiEval.ber.set_trun(threshold_run = 1.0)
```

Selects the threshold run value for BER tests. This value is the maximum bit error rate in any burst considered for the BER measurement.

param threshold_run numeric Range: 0 % to 100 %, Unit: %

set_tstart(*threshold_start: float*) → None

```
# SCPI: CONFigure:GSM:MEASurement<Instance>:MEValuation:BER:TSTart
driver.configure.multiEval.ber.set_tstart(threshold_start = 1.0)
```

Selects the threshold start value for BER tests. This value is the maximum bit error rate in the first burst of the BER measurement.

param threshold_start numeric Range: 0 % to 100 %, Unit: %

7.3 MultiEval

SCPI Commands

```
INITiate:GSM:MEASurement<Instance>:MEValuation
STOP:GSM:MEASurement<Instance>:MEValuation
ABORt:GSM:MEASurement<Instance>:MEValuation
```

class MultiEval

MultiEval commands group definition. 236 total commands, 9 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:GSM:MEASurement<Instance>:MEValuation
driver.multiEval.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:GSM:MEASurement<Instance>:MEValuation
driver.multiEval.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwGsmMeas.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:GSM:MEASurement<Instance>:MEValuation
driver.multiEval.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:GSM:MEASurement<Instance>:MEValuation
driver.multiEval.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwGsmMeas.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:GSM:MEASurement<Instance>:MEValuation
driver.multiEval.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
```

(continues on next page)

(continued from previous page)

```
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:GSM:MEASurement<Instance>:MEValuation
driver.multiEval.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwGsmMeas.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.clone()
```

Subgroups

7.3.1 State

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwGsmMeas.enums.ResourceState

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:STATe
value: enums.ResourceState = driver.multiEval.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_status: OFF | RUN | RDY OFF: measurement switched off, no resources allocated, no results available (when entered after ABORt...) RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued RDY: measurement has been terminated, valid results are available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.state.clone()
```

Subgroups

7.3.1.1 All

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RDY | RUN OFF: measurement switched off, no resources allocated, no results available (when entered after STOP...) RDY: measurement has been terminated, valid results are available RUN: measurement running (after INITiate..., READ...) , synchronization pending or adjusted, resources active or queued
- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: all necessary adjustments finished, measurement running ('adjusted') INV: not applicable because main_state: OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable because main_state: OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:STATe:ALL
value: FetchStruct = driver.multiEval.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.2 Trace

class Trace
Trace commands group definition. 32 total commands, 7 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.clone()
```

Subgroups

7.3.2.1 PowerVsTime

class PowerVsTime
PowerVsTime commands group definition. 8 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.powerVsTime.clone()
```

Subgroups

7.3.2.1.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
```

class Current
Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
value: List[float] = driver.multiEval.trace.powerVsTime.current.fetch()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:CURRent
value: List[float] = driver.multiEval.trace.powerVsTime.current.read()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

7.3.2.1.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
value: List[float] = driver.multiEval.trace.powerVsTime.average.fetch()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:AVERage
value: List[float] = driver.multiEval.trace.powerVsTime.average.read()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

7.3.2.1.3 Minimum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
```

class Minimum

Minimum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
value: List[float] = driver.multiEval.trace.powerVsTime.minimum.fetch()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MINimum
value: List[float] = driver.multiEval.trace.powerVsTime.minimum.read()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

7.3.2.1.4 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
value: List[float] = driver.multiEval.trace.powerVsTime.maximum.fetch()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PVTime:MAXimum
value: List[float] = driver.multiEval.trace.powerVsTime.maximum.read()
```

Returns the values of the power vs. time traces. 16 results are available for each symbol period of the measured slots (method RsCmwGsmMeas.Configure.MultiEval.mslots) . The trace covers 18.25 symbol periods before the beginning of the first slot in the measured slot range, 10 symbol periods after the end of the last measured slot. The length of the trace is given as: The first sample of the 'Measurement Slot' is at position m in the trace, where: The results of the current, average minimum and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float Range: -100 dB to 100 dB, Unit: dBm

7.3.2.2 EvMagnitude

class EvMagnitude

EvMagnitude commands group definition. 6 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.evMagnitude.clone()
```

Subgroups

7.3.2.2.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CURRent
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CURRent
value: List[float] = driver.multiEval.trace.evMagnitude.current.fetch()
```

Returns the values of the EVM traces. The results of the current, average and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n EVM results, depending on the burst and modulation type 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: 0 % to 100 %, Unit: %

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:CURRent
value: List[float] = driver.multiEval.trace.evMagnitude.current.read()
```

Returns the values of the EVM traces. The results of the current, average and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n EVM results, depending on the burst and modulation type 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: 0 % to 100 %, Unit: %

7.3.2.2.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:AVERage
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:AVERage
value: List[float] = driver.multiEval.trace.evMagnitude.average.fetch()
```

Returns the values of the EVM traces. The results of the current, average and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n EVM results, depending on the burst and modulation type 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: 0 % to 100 %, Unit: %

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:AVERage
value: List[float] = driver.multiEval.trace.evMagnitude.average.read()
```

Returns the values of the EVM traces. The results of the current, average and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n EVM results, depending on the burst and modulation type 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: 0 % to 100 %, Unit: %

7.3.2.2.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:MAXimum
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:MAXimum
value: List[float] = driver.multiEval.trace.evMagnitude.maximum.fetch()
```

Returns the values of the EVM traces. The results of the current, average and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n EVM results, depending on the burst and modulation type 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: 0 % to 100 %, Unit: %

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:EVMagnitude:MAXimum
value: List[float] = driver.multiEval.trace.evMagnitude.maximum.read()
```

Returns the values of the EVM traces. The results of the current, average and maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n EVM results, depending on the burst and modulation type 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: 0 % to 100 %, Unit: %

7.3.2.3 Merror

class Merror

Merror commands group definition. 6 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.merror.clone()
```

Subgroups

7.3.2.3.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:CURRent
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:CURRent
value: List[float] = driver.multiEval.trace.merror.current.fetch()
```

Returns the values of the magnitude error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n magnitude error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -100 % to 100 %, Unit: %

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:CURRent
value: List[float] = driver.multiEval.trace.merror.current.read()
```

Returns the values of the magnitude error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n magnitude error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -100 % to 100 %, Unit: %

7.3.2.3.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:AVERage
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:AVERage
value: List[float] = driver.multiEval.trace.merror.average.fetch()
```

Returns the values of the magnitude error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n magnitude error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to

symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -100 % to 100 %, Unit: %

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:AVERage
value: List[float] = driver.multiEval.trace.merror.average.read()
```

Returns the values of the magnitude error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n magnitude error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -100 % to 100 %, Unit: %

7.3.2.3.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:MAXimum
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:MAXimum
value: List[float] = driver.multiEval.trace.merror.maximum.fetch()
```

Returns the values of the magnitude error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n magnitude error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -100 % to 100 %, Unit: %

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:MERRor:MAXimum
value: List[float] = driver.multiEval.trace.merror.maximum.read()
```

Returns the values of the magnitude error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n magnitude error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -100 % to 100 %, Unit: %

7.3.2.4 Perror

class Perror

Perror commands group definition. 6 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.perror.clone()
```

Subgroups

7.3.2.4.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:CURRent
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:CURRent
value: List[float] = driver.multiEval.trace.perror.current.fetch()
```

Returns the values of the phase error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n phase error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -180 deg to 180 deg, Unit: deg

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:CURRent
value: List[float] = driver.multiEval.trace.perror.current.read()
```

Returns the values of the phase error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n phase error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -180 deg to 180 deg, Unit: deg

7.3.2.4.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:AVERage
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:AVERage
value: List[float] = driver.multiEval.trace.perror.average.fetch()
```

Returns the values of the phase error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n phase error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -180 deg to 180 deg, Unit: deg

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:AVERage
value: List[float] = driver.multiEval.trace.perror.average.read()
```

Returns the values of the phase error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n phase error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -180 deg to 180 deg, Unit: deg

7.3.2.4.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:MAXimum
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:MAXimum
value: List[float] = driver.multiEval.trace.perror.maximum.fetch()
```

Returns the values of the phase error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n phase error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -180 deg to 180 deg, Unit: deg

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:PERRor:MAXimum
value: List[float] = driver.multiEval.trace.perror.maximum.read()
```

Returns the values of the phase error traces. The results of the current, average and minimum/maximum traces can be retrieved.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n phase error results, depending on the type of modulation 8PSK/16-QAM modulation: 142 values (one value per symbol period, symbol 3 to symbol 144) GMSK modulation: 588 values (four values per symbol period, symbol 0.5 to symbol 147.5) Access burst: 348 values (four values per symbol period, symbol 0.5 to symbol 87.5) Range: -180 deg to 180 deg, Unit: deg

7.3.2.5 Smodulation

class Smodulation

Smodulation commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.smodulation.clone()
```

Subgroups

7.3.2.5.1 Time

class Time
Time commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.smodulation.time.clone()
```

Subgroups

7.3.2.5.1.1 Current

SCPI Commands

```
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:SMODulation:TIME:CURRent
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:SMODulation:TIME:CURRent
```

class Current
Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:TRACe:SMODulation:TIME[:CURRent]
value: List[float] = driver.multiEval.trace.smodulation.time.current.fetch()
```

Returns the spectrum due to modulation trace values measured at a selected offset frequency (method RsCmwGsmMeas. Configure.MultiEval.Smodulation.tdfSelect) .

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n power results, 4 for each symbol period of the 'Measured Slot'
Range: -100 dB to 100 dB, Unit: dB

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>
↪:MEValuation:TRACe:SMODulation:TIME[:CURRent]
value: List[float] = driver.multiEval.trace.smodulation.time.current.read()
```

Returns the spectrum due to modulation trace values measured at a selected offset frequency (method RsCmwGsmMeas. Configure.MultiEval.Smodulation.tdfSelect) .

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n power results, 4 for each symbol period of the 'Measured Slot' Range: -100 dB to 100 dB, Unit: dB

7.3.2.6 Sswitching

class Sswitching

Sswitching commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.sswitching.clone()
```

Subgroups

7.3.2.6.1 Time

class Time

Time commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.sswitching.time.clone()
```

Subgroups

7.3.2.6.1.1 Current

SCPI Commands

```
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:SSWitching:TIME:CURRent
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:SSWitching:TIME:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:TRACe:SSWitching:TIME[:CURRent]
value: List[float] = driver.multiEval.trace.sswitching.time.current.fetch()
```

Returns the spectrum due to switching trace values measured at a selected offset frequency (method RsCmwGsmMeas.Configure. MultiEval.Sswitching.tdfSelect) .

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n power results, 4 for each symbol period of all measured slots
Range: -100 dBm to 55 dBm, Unit: dBm

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>
↪:MEValuation:TRACe:SSWitching:TIME[:CURRent]
value: List[float] = driver.multiEval.trace.sswitching.time.current.read()
```

Returns the spectrum due to switching trace values measured at a selected offset frequency (method RsCmwGsmMeas.Configure. MultiEval.Sswitching.tdfSelect) .

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return results: float n power results, 4 for each symbol period of all measured slots
Range: -100 dBm to 55 dBm, Unit: dBm

7.3.2.7 Iq

class Iq

Iq commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.trace.iq.clone()
```

Subgroups

7.3.2.7.1 Current

SCPI Commands

```
READ:GSM:MEASurement<Instance>:MEValuation:TRACe:IQ:CURRent
FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:IQ:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Iphase: List[float]: No parameter help available
- Qphase: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:TRACe:IQ[:CURRent]
value: ResultData = driver.multiEval.trace.iq.current.fetch()
```

Returns the results in the I/Q constellation diagram.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:TRACe:IQ[:CURRent]
value: ResultData = driver.multiEval.trace.iq.current.read()
```

Returns the results in the I/Q constellation diagram.

return structure: for return value, see the help for ResultData structure arguments.

7.3.3 MvThroughput

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:MVTHroughput
```

class MvThroughput

MvThroughput commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → float

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:MVTHroughput
value: float = driver.multiEval.mvThroughput.fetch()
```

Returns the modulation view throughput, i.e. the percentage of measurement intervals where the detected burst pattern was found to correspond to the 'Modulation View' settings. See also 'Mod. View Throughput'

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mv_throughput: float Modulation view throughput Range: 0 % to 100 %, Unit: %

7.3.4 PowerVsTime

class PowerVsTime

PowerVsTime commands group definition. 10 total commands, 8 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.powerVsTime.clone()
```

Subgroups

7.3.4.1 All

SCPI Commands

```
CALCulate:GSM:MEASurement<Instance>:MEValuation:PVTime:ALL
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:ALL
READ:GSM:MEASurement<Instance>:MEValuation:PVTime:ALL
```

class All

All commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Bursts_Out_Tol: float or bool: float Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:PVTime CMDLINK]) exceeding the specified limits, see 'Limits (Power vs. Time) ' Range: 0 % to 100 %, Unit: %
- Avg_Burst_Pow_Avg: List[float]: No parameter help available
- Avg_Burst_Pow_Cur: List[float]: No parameter help available
- Max_Burst_Pow_Cur: List[float]: No parameter help available
- Min_Burst_Pow_Cur: List[float]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Bursts_Out_Tol: float or bool: float Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:PVTime CMDLINK]) exceeding the specified limits, see 'Limits (Power vs. Time) ' Range: 0 % to 100 %, Unit: %
- Avg_Burst_Pow_Avg: List[float or bool]: No parameter help available
- Avg_Burst_Pow_Cur: List[float or bool]: No parameter help available
- Max_Burst_Pow_Cur: List[float or bool]: No parameter help available
- Min_Burst_Pow_Cur: List[float or bool]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:PVTime[:ALL]
value: CalculateStruct = driver.multiEval.powerVsTime.all.calculate()
```

Returns burst power values for slot 0 to slot 7. In addition to the current value statistical values are returned (average, minimum and maximum) . The relative number of bursts out of tolerance is also returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime[:ALL]
value: ResultData = driver.multiEval.powerVsTime.all.fetch()
```

Returns burst power values for slot 0 to slot 7. In addition to the current value statistical values are returned (average, minimum and maximum) . The relative number of bursts out of tolerance is also returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:PVTime[:ALL]
value: ResultData = driver.multiEval.powerVsTime.all.read()
```

Returns burst power values for slot 0 to slot 7. In addition to the current value statistical values are returned (average, minimum and maximum) . The relative number of bursts out of tolerance is also returned. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for ResultData structure arguments.

7.3.4.2 Tsc

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:TSC
```

class Tsc

Tsc commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[RsCmwGsmMeas.enums.TscB]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:TSC
value: List[enums.TscB] = driver.multiEval.powerVsTime.tsc.fetch()
```

Returns the detected training sequence code (TSC) and burst type for all measured timeslots. 8 values are returned, irrespective of the 'No. of Slots' measured (method RsCmwGsmMeas.Configure.MultiEval.mslots) . If 'No. of Slots' < 8, some of the returned values are NAN.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return tsc: OFF | NB0 | NB1 | NB2 | NB3 | NB4 | NB5 | NB6 | NB7 | DUMMy | AB0 | AB1 | AB2 | AB3 | AB4 | AB5 | AB6 | AB7 Detected TSC (8 values) : OFF: Inactive slot NB0 … NB7: Normal burst, training sequence TSC0 to TSC7 DUMMY: Dummy burst AB0 … AB7: Access burst, TSC 0 to TSC7

7.3.4.3 Btype

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:BTYPe
```

class Btype

Btype commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[RsCmwGsmMeas.enums.SlotInfo]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:BTYPe
value: List[enums.SlotInfo] = driver.multiEval.powerVsTime.btype.fetch()
```

Returns the detected burst type for all measured timeslots. 8 values are returned, irrespective of the 'No. of Slots' measured (method RsCmwGsmMeas.Configure.MultiEval.mslots) . If 'No. of Slots' < 8, some of the returned values are NAN.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_type: OFF | GMSK | EPSK | ACCess | Q16 Detected burst type (8 values) : GMSK: Normal burst, GMSK-modulated EPSK: Normal burst, 8PSK-modulated ACCess: Access burst Q16: Normal burst, 16-QAM-modulated OFF: Inactive slot

7.3.4.4 RsTiming

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:RSTiming
```

class RsTiming

RsTiming commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:RSTiming
value: List[float] = driver.multiEval.powerVsTime.rsTiming.fetch()
```

Returns the slot timing for all measured timeslots, relative to the timing of the 'Measurement Slot'. The relative slot timing of the 'Measurement Slot' is always zero. The relative slot timing of the other timeslots is the deviation of the measured relative timing from the nominal timing. The nominal timing is based on a timeslot length of 156. 25 symbol durations. The command returns 8 values, irrespective of the 'No. of Slots' measured (method RsCmwGsmMeas. Configure.MultiEval.mslots) . If 'No. of Slots' < 8, some of the returned values are NAN.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return rel_slot_timing: float Range: -1500 Sym to 1500 Sym, Unit: bits

7.3.4.5 Current

class Current
Current commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.powerVsTime.current.clone()
```

Subgroups

7.3.4.5.1 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:CURRent:SVECtor
```

class Svector
Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct
Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:CURRent:SVECtor
value: FetchStruct = driver.multiEval.powerVsTime.current.svector.fetch()
```

Returns special burst power values for the 'Measure Slot'. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.4.6 Average

class Average
Average commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.powerVsTime.average.clone()
```

Subgroups

7.3.4.6.1 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:AVERage:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:AVERage:SVECtor
value: FetchStruct = driver.multiEval.powerVsTime.average.svector.fetch()
```

Returns special burst power values for the 'Measure Slot'. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.4.7 Minimum

class Minimum

Minimum commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.powerVsTime.minimum.clone()
```

Subgroups

7.3.4.7.1 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:MINimum:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:MINimum:SVECtor
value: FetchStruct = driver.multiEval.powerVsTime.minimum.svector.fetch()
```

Returns special burst power values for the 'Measure Slot'. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.4.8 Maximum

class Maximum

Maximum commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.powerVsTime.maximum.clone()
```

Subgroups

7.3.4.8.1 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:MAXimum:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct
Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:PVTime:MAXimum:SVECtor
value: FetchStruct = driver.multiEval.powerVsTime.maximum.svector.fetch()
```

Returns special burst power values for the 'Measure Slot'. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.5 Modulation

class Modulation
Modulation commands group definition. 15 total commands, 6 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.modulation.clone()
```

Subgroups

7.3.5.1 Average

SCPI Commands

```
CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:AVERage
FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:AVERage
READ:GSM:MEASurement<Instance>:MEValuation:MODulation:AVERage
```

class Average
Average commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %

- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error RMS and peak value Range: 0 % to 100 %, Unit: %
- Phase_Error_Rms: float: float Phase error RMS and peak value Range: 0 deg to 180 deg, Unit: deg
- Phase_Error_Peak: float: float Phase error RMS and peak value Range: 0 deg to 180 deg, Unit: deg
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Sym to 100 Sym, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AMPM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error RMS and peak value Range: 0 % to 100 %, Unit: %
- Phase_Error_Rms: float: float Phase error RMS and peak value Range: 0 deg to 180 deg, Unit: deg
- Phase_Error_Peak: float: float Phase error RMS and peak value Range: 0 deg to 180 deg, Unit: deg
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Sym to 100 Sym, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AMPM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: CalculateStruct = driver.multiEval.modulation.average.calculate()
```

Returns the average single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.multiEval.modulation.average.fetch()
```

Returns the average single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.multiEval.modulation.average.read()
```

Returns the average single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

7.3.5.2 Current

SCPI Commands

```
CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:CURRent
FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:CURRent
READ:GSM:MEASurement<Instance>:MEValuation:MODulation:CURRent
```

class Current

Current commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm_Rms: enums.ResultStatus2: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: enums.ResultStatus2: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: enums.ResultStatus2: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: enums.ResultStatus2: float Magnitude error peak value Range: -100 % to 100 %, Unit: %
- Phase_Error_Rms: enums.ResultStatus2: float Phase error RMS value Range: 0 deg to 180 deg, Unit: deg

- Phase_Error_Peak: enums.ResultStatus2: float Phase error peak value Range: -180 deg to 180 deg, Unit: deg
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Sym to 100 Sym, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 %, Unit: %
- Phase_Error_Rms: float: float Phase error RMS value Range: 0 deg to 180 deg, Unit: deg
- Phase_Error_Peak: float: float Phase error peak value Range: -180 deg to 180 deg, Unit: deg
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Sym to 100 Sym, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: CalculateStruct = driver.multiEval.modulation.current.calculate()
```

Returns the current and minimum/maximum single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.multiEval.modulation.current.fetch()
```

Returns the current and minimum/maximum single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.multiEval.modulation.current.read()
```

Returns the current and minimum/maximum single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

7.3.5.3 Maximum

SCPI Commands

```
CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:MAXimum
FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:MAXimum
READ:GSM:MEASurement<Instance>:MEValuation:MODulation:MAXimum
```

class Maximum

Maximum commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm_Rms: enums.ResultStatus2: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: enums.ResultStatus2: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: enums.ResultStatus2: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: enums.ResultStatus2: float Magnitude error peak value Range: -100 % to 100 %, Unit: %
- Phase_Error_Rms: enums.ResultStatus2: float Phase error RMS value Range: 0 deg to 180 deg, Unit: deg
- Phase_Error_Peak: enums.ResultStatus2: float Phase error peak value Range: -180 deg to 180 deg, Unit: deg

- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Sym to 100 Sym, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 %, Unit: %
- Phase_Error_Rms: float: float Phase error RMS value Range: 0 deg to 180 deg, Unit: deg
- Phase_Error_Peak: float: float Phase error peak value Range: -180 deg to 180 deg, Unit: deg
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Sym to 100 Sym, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:MAXimum
value: CalculateStruct = driver.multiEval.modulation.maximum.calculate()
```

Returns the current and minimum/maximum single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:MAXimum
value: ResultData = driver.multiEval.modulation.maximum.fetch()
```

Returns the current and minimum/maximum single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:MODulation:MAXimum
value: ResultData = driver.multiEval.modulation.maximum.read()
```

Returns the current and minimum/maximum single slot modulation results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

7.3.5.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:SDEViation
READ:GSM:MEASurement<Instance>:MEValuation:MODulation:SDEViation
```

class StandardDev

StandardDev commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:SMODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 50 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 50 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS and peak value Range: 0 % to 50 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error RMS and peak value Range: 0 % to 50 %, Unit: %
- Phase_Error_Rms: float: float Phase error RMS and peak value Range: 0 deg to 90 deg, Unit: deg
- Phase_Error_Peak: float: float Phase error RMS and peak value Range: 0 deg to 90 deg, Unit: deg
- Iq_Offset: float: float I/Q origin offset Range: 0 dB to 50 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: 0 dB to 50 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: 0 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: 0 Sym to 100 Sym, Unit: Symbol
- Burst_Power: float: float Burst power Range: 0 dB to 71 dB, Unit: dB
- Am_Pm_Delay: float: float AMPM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: 0 s to 0.9225E-6 s, Unit: s

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.multiEval.modulation.standardDev.fetch()
```

Returns the standard deviation of the single slot modulation results of the multi-evaluation measurement. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.multiEval.modulation.standardDev.read()
```

Returns the standard deviation of the single slot modulation results of the multi-evaluation measurement. The number to the left of each result parameter is provided for easy identification of the parameter position within the result array.

return structure: for return value, see the help for ResultData structure arguments.

7.3.5.5 Percentile

SCPI Commands

```
CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:PERCentile
FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:PERCentile
READ:GSM:MEASurement<Instance>:MEValuation:MODulation:PERCentile
```

class Percentile

Percentile commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %
- Evm: enums.ResultStatus2: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Magnitude_Error: enums.ResultStatus2: float Magnitude error percentile Range: 0 % to 100 %, Unit: %
- Phase_Error: enums.ResultStatus2: float Phase error percentile Range: 0 deg to 180 deg, Unit: deg

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tolerance: int: decimal Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:MODulation CMDLINK]) exceeding the specified modulation limits. Range: 0 % to 100 %, Unit: %

- Evm: float: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Magnitude_Error: float: float Magnitude error percentile Range: 0 % to 100 %, Unit: %
- Phase_Error: float: float Phase error percentile Range: 0 deg to 180 deg, Unit: deg

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:MODulation:PERCentile
value: CalculateStruct = driver.multiEval.modulation.percentile.calculate()
```

Returns the 95th percentile results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:PERCentile
value: ResultData = driver.multiEval.modulation.percentile.fetch()
```

Returns the 95th percentile results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:MODulation:PERCentile
value: ResultData = driver.multiEval.modulation.percentile.read()
```

Returns the 95th percentile results of the multi-evaluation measurement. The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for ResultData structure arguments.

7.3.5.6 Dbits

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:DBITs
```

class Dbits

Dbits commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:MODulation:DBITs
value: List[int] = driver.multiEval.modulation.dbits.fetch()
```

Returns the demodulated bits of the 'Measurement Slot'. For GMSK modulation, a symbol consists of 1 bit, for 8PSK of 3 bits, for 16-QAM of 4 bits.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return demod_bits: decimal 142 values, one value per symbol, representing the demodulated bits of the symbol in decimal presentation Range: 0 to 15

7.3.6 Sswitching

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:SSWitching
READ:GSM:MEASurement<Instance>:MEValuation:SSWitching
```

class Sswitching

Sswitching commands group definition. 7 total commands, 1 Sub-groups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tol_Count: float: float Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:SSWitching CMDLINK]) exceeding the specified limits, see 'Limits (Spectrum Switching) ' Range: 0 % to 100 %, Unit: %
- Carrier_Power: float: float Measured carrier output power (reference power) Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:SSWitching
value: ResultData = driver.multiEval.sswitching.fetch()
```

Returns general spectrum switching results.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:SSWitching
value: ResultData = driver.multiEval.sswitching.read()
```

Returns general spectrum switching results.

return structure: for return value, see the help for ResultData structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.sswitching.clone()
```

Subgroups

7.3.6.1 Frequency

SCPI Commands

```
CALCulate:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency
FETCh:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency
READ:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency
```

class Frequency

Frequency commands group definition. 5 total commands, 1 Sub-groups, 3 group commands

calculate() → List[RsCmwGsmMeas.enums.ResultStatus2]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency
value: List[enums.ResultStatus2] = driver.multiEval.sswitching.frequency.
↪calculate()
```

Returns the maximum burst power measured at a series of frequencies. The frequencies are determined by the offset values defined via the command method RsCmwGsmMeas.Configure.MultiEval.Sswitching.ofrequence. All defined offset values are considered (irrespective of their activation status) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return power: No help available

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency
value: List[float] = driver.multiEval.sswitching.frequency.fetch()
```

Returns the maximum burst power measured at a series of frequencies. The frequencies are determined by the offset values defined via the command method RsCmwGsmMeas.Configure.MultiEval.Sswitching.ofrequence. All defined offset values are considered (irrespective of their activation status) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return power: No help available

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency
value: List[float] = driver.multiEval.sswitching.frequency.read()
```

Returns the maximum burst power measured at a series of frequencies. The frequencies are determined by the offset values defined via the command method RsCmwGsmMeas.Configure.MultiEval.Sswitching.ofrequence. All defined offset values are considered (irrespective of their activation status) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return power: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.sswitching.frequency.clone()
```

Subgroups

7.3.6.1.1 Limits

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency:LIMits
READ:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency:LIMits
```

class Limits

Limits commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency:LIMits
value: List[float] = driver.multiEval.sswitching.frequency.limits.fetch()
```

Queries the limits for spectrum switching frequency. See also method RsCmwGsmMeas.Configure.MultiEval.Sswitching. ofrequence.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return limit: float 41 values display limits at the following frequency offsets: values 1 to 20 = minus offset 19 to minus offset 0 value 21 = carrier frequency, no offset values 21 to 41 = plus offset 0 to plus offset 19 Unit: dBm

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:SSWitching:FREQuency:LIMits
value: List[float] = driver.multiEval.sswitching.frequency.limits.read()
```

Queries the limits for spectrum switching frequency. See also method RsCmwGsmMeas.Configure.MultiEval.Sswitching. ofrequence.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return limit: float 41 values display limits at the following frequency offsets: values 1 to 20 = minus offset 19 to minus offset 0 value 21 = carrier frequency, no offset values 21 to 41 = plus offset 0 to plus offset 19 Unit: dBm

7.3.7 Smodulation

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:SMODulation
READ:GSM:MEASurement<Instance>:MEValuation:SMODulation
```

class Smodulation

Smodulation commands group definition. 7 total commands, 1 Sub-groups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Out_Of_Tol_Count: float: float Percentage of measurement intervals / bursts of the statistic count ([CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:SMODulation CMDLINK]) exceeding the specified limits (see 'Limits (Spectrum Modulation) ') . Range: 0 % to 100 %, Unit: %
- Carrier_Power: float: float Measured carrier output power (reference power) Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → ResultData

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:SMODulation
value: ResultData = driver.multiEval.smodulation.fetch()
```

Returns general spectrum modulation results.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:SMODulation
value: ResultData = driver.multiEval.smodulation.read()
```

Returns general spectrum modulation results.

return structure: for return value, see the help for ResultData structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.smodulation.clone()
```

Subgroups

7.3.7.1 Frequency

SCPI Commands

```
CALCulate:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency
FETCh:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency
READ:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency
```

class Frequency

Frequency commands group definition. 5 total commands, 1 Sub-groups, 3 group commands

calculate() → List[RsCmwGsmMeas.enums.ResultStatus2]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency
value: List[enums.ResultStatus2] = driver.multiEval.smodulation.frequency.
↪calculate()
```

Returns the average burst power measured at a series of frequencies. The frequencies are determined by the offset values defined via the command method RsCmwGsmMeas.Configure.MultiEval.Smodulation.ofrequence. All defined offset values are considered (irrespective of their activation status) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return power: No help available

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency
value: List[float] = driver.multiEval.smodulation.frequency.fetch()
```

Returns the average burst power measured at a series of frequencies. The frequencies are determined by the offset values defined via the command method RsCmwGsmMeas.Configure.MultiEval.Smodulation.ofrequence. All defined offset values are considered (irrespective of their activation status) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return power: No help available

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency
value: List[float] = driver.multiEval.smodulation.frequency.read()
```

Returns the average burst power measured at a series of frequencies. The frequencies are determined by the offset values defined via the command method RsCmwGsmMeas.Configure.MultiEval.Smodulation.ofrequence. All defined offset values are considered (irrespective of their activation status) . The values described below are returned by FETCh and READ commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return power: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.smodulation.frequency.clone()
```

Subgroups

7.3.7.1.1 Limits

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency:LIMits
READ:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency:LIMits
```

class Limits

Limits commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency:LIMits
value: List[float] = driver.multiEval.smodulation.frequency.limits.fetch()
```

Queries the limits for spectrum modulation frequency. See also method RsCmwGsmMeas.Configure.MultiEval.Smodulation. ofrequence

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return limit: float 41 values display limits at the following frequency offsets: values 1 to 20 = minus offset 19 to minus offset 0 value 21 = carrier frequency, no offset values 21 to 41 = plus offset 0 to plus offset 19 Unit: dB

read() → List[float]

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:SMODulation:FREQuency:LIMits
value: List[float] = driver.multiEval.smodulation.frequency.limits.read()
```

Queries the limits for spectrum modulation frequency. See also method RsCmwGsmMeas.Configure.MultiEval.Smodulation. ofrequence

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return limit: float 41 values display limits at the following frequency offsets: values 1 to 20 = minus offset 19 to minus offset 0 value 21 = carrier frequency, no offset values 21 to 41 = plus offset 0 to plus offset 19 Unit: dB

7.3.8 Ber

SCPI Commands

```
READ:GSM:MEASurement<Instance>:MEValuation:BER
FETCh:GSM:MEASurement<Instance>:MEValuation:BER
```

class Ber

Ber commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Ber: float or bool: float % bit error rate Range: 0 % to 100 %, Unit: %
- Ber_Absolute: int or bool: float Total number of detected bit errors The BER measurement evaluates 114 data bits per GMSK-modulated normal burst, 306 data bits per 8PSK-modulated burst. Range: 0 to no. of measured bits
- Ber_Count: int or bool: float Total number of evaluated bits Range: 0 to no. of measured bits

class ReadStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Ber: float or bool: float % bit error rate Range: 0 % to 100 %, Unit: %
- Ber_Absolute: List[float or bool]: float Total number of detected bit errors The BER measurement evaluates 114 data bits per GMSK-modulated normal burst, 306 data bits per 8PSK-modulated burst. Range: 0 to no. of measured bits
- Ber_Count: List[float or bool]: float Total number of evaluated bits Range: 0 to no. of measured bits

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:BER
value: FetchStruct = driver.multiEval.ber.fetch()
```

Returns the measured bit error rate. The BER measurement must be enabled using method RsCmwGsmMeas.Configure.MultiEval. Result.ber.

return structure: for return value, see the help for FetchStruct structure arguments.

read() → ReadStruct

```
# SCPI: READ:GSM:MEASurement<Instance>:MEValuation:BER
value: ReadStruct = driver.multiEval.ber.read()
```

Returns the measured bit error rate. The BER measurement must be enabled using method RsCmwGsmMeas.Configure.MultiEval. Result.ber.

return structure: for return value, see the help for ReadStruct structure arguments.

7.3.9 ListPy

class ListPy
ListPy commands group definition. 157 total commands, 8 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.clone()
```

Subgroups

7.3.9.1 Sreliability

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SRELiability
```

class Sreliability
Sreliability commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SRELiability
value: List[int] = driver.multiEval.listPy.sreliability.fetch()
```

Returns the segment reliability for all measured list mode segments. A common reliability indicator of zero indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments. If you get a non-zero common reliability indicator, you can use this command to retrieve the individual reliability values of all measured segments for further analysis.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return seg_reliability: decimal Comma-separated list of values, one per measured segment The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.

7.3.9.2 PowerVsTime

class PowerVsTime
PowerVsTime commands group definition. 21 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.powerVsTime.clone()
```

Subgroups

7.3.9.2.1 AbPower

class AbPower
AbPower commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.powerVsTime.abPower.clone()
```

Subgroups

7.3.9.2.1.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:ABPower:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:ABPower:CURRent
```

class Current
Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:ABPower:CURRent
value: List[float] = driver.multiEval.listPy.powerVsTime.abPower.current.
↪calculate()
```

Return average burst power results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return average_burst_pow: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:ABPower:CURRent
value: List[float] = driver.multiEval.listPy.powerVsTime.abPower.current.fetch()
```

Return average burst power results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return average_burst_pow: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.2.1.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:ABPower:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:ABPower:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:ABPower:AVERage
value: List[float] = driver.multiEval.listPy.powerVsTime.abPower.average.
↪calculate()
```

Return average burst power results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return average_burst_pow: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:ABPower:AVERage
value: List[float] = driver.multiEval.listPy.powerVsTime.abPower.average.fetch()
```

Return average burst power results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return average_burst_pow: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.2.2 Svector

class Svector
Svector commands group definition. 12 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.powerVsTime.svector.clone()
```

Subgroups

7.3.9.2.2.1 Uminimum

class Uminimum
Uminimum commands group definition. 4 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.powerVsTime.svector.uminimum.clone()
```

Subgroups

7.3.9.2.2.2 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMINimum:CURRent
```

class Current
Current commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMINimum:CURRent
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.uminimum.
↪current.fetch()
```

Return minimum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_min: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.3 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMINimum:AVERage
```

class Average

Average commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMINimum:AVERage
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.uminimum.
↪average.fetch()
```

Return minimum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_min: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.4 Minimum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMINimum:MINimum
```

class Minimum

Minimum commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMINimum:MINimum
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.uminimum.
↪minimum.fetch()
```

Return minimum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_min: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.5 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMINimum:MAXimum
```

class Maximum

Maximum commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMINimum:MAXimum
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.uminimum.
↪maximum.fetch()
```

Return minimum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_min: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.6 Umaximum

class Umaximum

Umaximum commands group definition. 4 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.powerVsTime.svector.umaximum.clone()
```

Subgroups

7.3.9.2.2.7 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMAXimum:CURRent
```

class Current

Current commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMAXimum:CURRent
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.umaximum.
↪current.fetch()
```

Return maximum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_max: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.8 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMAXimum:AVERage
```

class Average

Average commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMAXimum:AVERage
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.umaximum.
↪average.fetch()
```

Return maximum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_max: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.9 Minimum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMAXimum:MINimum
```

class Minimum

Minimum commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMAXimum:MINimum
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.umaximum.
↪minimum.fetch()
```

Return maximum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_max: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.10 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:UMAXimum:MAXimum
```

class Maximum

Maximum commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:UMAXimum:MAXimum
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.umaximum.
↪maximum.fetch()
```

Return maximum power across the useful part of the burst for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return usefull_part_max: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.11 Subvector<SubVector>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.multiEval.listPy.powerVsTime.svector.subvector.repcap_subVector_get()
driver.multiEval.listPy.powerVsTime.svector.subvector.repcap_subVector_set(repcap.
↪SubVector.Nr1)
```

class Subvector

Subvector commands group definition. 4 total commands, 4 Sub-groups, 0 group commands Repeated Capability: SubVector, default value after init: SubVector.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.powerVsTime.svector.subvector.clone()
```

Subgroups

7.3.9.2.2.12 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:SUBVector<SubVector>
↪:CURRent
```

class Current

Current commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch(*subVector*=<*SubVector.Default: -1*>) → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:SUBVector<nr>:CURRent
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.subvector.
↪current.fetch(subVector = repcap.SubVector.Default)
```

Return burst power at a specific burst position for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param subVector optional repeated capability selector. Default value: Nr1 (settable in the interface 'Subvector')

return subvector: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.13 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:SUBVector<SubVector>
↪:AVERage
```

class Average

Average commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch(*subVector*=<*SubVector.Default: -1*>) → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:SUBVector<nr>:AVERage
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.subvector.
↪average.fetch(subVector = repcap.SubVector.Default)
```

Return burst power at a specific burst position for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param subVector optional repeated capability selector. Default value: Nr1 (settable in the interface 'Subvector')

return subvector: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.14 Minimum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:SUBVector<SubVector>
↪:MINimum
```

class Minimum

Minimum commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch(*subVector*=<*SubVector.Default: -1*>) → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:SUBVector<nr>:MINimum
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.subvector.
↪minimum.fetch(subVector = repcap.SubVector.Default)
```

Return burst power at a specific burst position for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param subVector optional repeated capability selector. Default value: Nr1 (settable in the interface 'Subvector')

return subvector: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.2.15 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:SVECtor:SUBVector<SubVector>
↪:MAXimum
```

class Maximum

Maximum commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch(*subVector*=<*SubVector.Default: -1*>) → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:PVTime:SVECtor:SUBVector<nr>:MAXimum
value: List[float] = driver.multiEval.listPy.powerVsTime.svector.subvector.
↪maximum.fetch(subVector = repcap.SubVector.Default)
```

Return burst power at a specific burst position for all measured list mode segments.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param subVector optional repeated capability selector. Default value: Nr1 (settable in the interface 'Subvector')

return subvector: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

7.3.9.2.3 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: List[float]: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: List[float]: No parameter help available

calculate(*segment_start: Optional[int] = None*, *segment_count: Optional[int] = None*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:AVERage
value: CalculateStruct = driver.multiEval.listPy.powerVsTime.average.
↪calculate(segment_start = 1, segment_count = 1)
```

Returns the power vs. time results in list mode. By default results are returned for all measured segments. Use the optional parameters to query only a subset. The values listed below in curly brackets {}

are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment_start integer First segment to be returned

param segment_count integer Number of segments to be returned

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment_start: Optional[int] = None*, *segment_count: Optional[int] = None*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:AVERage
value: FetchStruct = driver.multiEval.listPy.powerVsTime.average.fetch(segment_
↪start = 1, segment_count = 1)
```

Returns the power vs. time results in list mode. By default results are returned for all measured segments. Use the optional parameters to query only a subset. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment_start integer First segment to be returned

param segment_count integer Number of segments to be returned

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.2.4 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:CURRent
```

class Current

Current commands group definition. 3 total commands, 1 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available

- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: List[float]: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: List[float]: No parameter help available

calculate(*segment_start: Optional[int] = None*, *segment_count: Optional[int] = None*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:CURRent
value: CalculateStruct = driver.multiEval.listPy.powerVsTime.current.
↪calculate(segment_start = 1, segment_count = 1)
```

Returns the power vs. time results in list mode. By default results are returned for all measured segments. Use the optional parameters to query only a subset. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment_start integer First segment to be returned

param segment_count integer Number of segments to be returned

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment_start: Optional[int] = None*, *segment_count: Optional[int] = None*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:CURRent
value: FetchStruct = driver.multiEval.listPy.powerVsTime.current.fetch(segment_
↪start = 1, segment_count = 1)
```

Returns the power vs. time results in list mode. By default results are returned for all measured segments. Use the optional parameters to query only a subset. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method

RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment_start integer First segment to be returned

param segment_count integer Number of segments to be returned

return structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.powerVsTime.current.clone()
```

Subgroups

7.3.9.2.4.1 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:CURRent:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Usefull_Part_Min: List[float]: No parameter help available
- Usefull_Part_Max: List[float]: No parameter help available
- Subvector_1: List[float]: No parameter help available
- Subvector_2: List[float]: No parameter help available
- Subvector_3: List[float]: No parameter help available
- Subvector_4: List[float]: No parameter help available

- Subvector_5: List[float]: No parameter help available
- Subvector_6: List[float]: No parameter help available
- Subvector_7: List[float]: No parameter help available
- Subvector_8: List[float]: No parameter help available
- Subvector_9: List[float]: No parameter help available
- Subvector_10: List[float]: No parameter help available
- Subvector_11: List[float]: No parameter help available
- Subvector_12: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:PVTime:CURRent:SVECtor
value: FetchStruct = driver.multiEval.listPy.powerVsTime.current.svector.fetch()
```

Returns special burst power values in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.3 Modulation

class Modulation

Modulation commands group definition. 99 total commands, 14 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.clone()
```

Subgroups

7.3.9.3.1 Evm

class Evm

Evm commands group definition. 16 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.clone()
```

Subgroups

7.3.9.3.1.1 Rms

class Rms
Rms commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.rms.clone()
```

Subgroups

7.3.9.3.1.2 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:CURRent
```

class Current
Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.current.
↪calculate()
```

Return error vector magnitude RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_rms: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.current.fetch()
```

Return error vector magnitude RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_rms: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

7.3.9.3.1.3 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.average.
↪calculate()
```

Return error vector magnitude RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_rms: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.average.fetch()
```

Return error vector magnitude RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_rms: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

7.3.9.3.1.4 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.maximum.
↪calculate()
```

Return error vector magnitude RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_rms: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.maximum.fetch()
```

Return error vector magnitude RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_rms: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

7.3.9.3.1.5 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.rms.standardDev.
↪fetch()
```

Return error vector magnitude RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_rms: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

7.3.9.3.1.6 Peak

class Peak

Peak commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.evm.peak.clone()
```

Subgroups

7.3.9.3.1.7 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.current.
↪calculate()
```

Return error vector magnitude peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_peak: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:CURRent
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.current.fetch()
```

Return error vector magnitude peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_peak: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

7.3.9.3.1.8 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.average.
↪calculate()
```

Return error vector magnitude peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_peak: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:AVERage
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.average.fetch()
```

Return error vector magnitude peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_peak: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

7.3.9.3.1.9 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.maximum.
↪calculate()
```

Return error vector magnitude peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_peak: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.maximum.fetch()
```

Return error vector magnitude peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_peak: float Comma-separated list of values, one per measured segment
Range: 0 % to 100 %, Unit: %

7.3.9.3.1.10 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.evm.peak.standardDev.
↪fetch()
```

Return error vector magnitude peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm_peak: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.3.1.11 Percentile

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PERCentile
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PERCentile
```

class Percentile

Percentile commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PERCentile
value: List[float] = driver.multiEval.listPy.modulation.evm.percentile.
↪calculate()
```

Return error vector magnitude 95th percentile values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PERCentile
value: List[float] = driver.multiEval.listPy.modulation.evm.percentile.fetch()
```

Return error vector magnitude 95th percentile values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return evm: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.3.2 Merror

class Merror

Merror commands group definition. 16 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.clone()
```

Subgroups

7.3.9.3.2.1 Rms

class Rms

Rms commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.rms.clone()
```

Subgroups

7.3.9.3.2.2 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.current.
↪calculate()
```

Return magnitude error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_rms: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.current.
↪fetch()
```

Return magnitude error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_rms: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.3.2.3 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.average.
↪calculate()
```

Return magnitude error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_rms: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.average.
↪fetch()
```

Return magnitude error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_rms: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.3.2.4 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.maximum.
↪calculate()
```

Return magnitude error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_rms: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.maximum.
↪fetch()
```

Return magnitude error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_rms: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.3.2.5 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.rms.standardDev.
↪fetch()
```

Return magnitude error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_rms: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.3.2.6 Peak

class Peak

Peak commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.merror.peak.clone()
```

Subgroups

7.3.9.3.2.7 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.current.
↪calculate()
```

Return magnitude error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_peak: float Comma-separated list of values, one per measured segment Range: -100 % to 100 % (AVERage: 0 % to 100 %, SDEViation: 0 % to 50 %) , Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:CURRent
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.current.
↪fetch()
```

Return magnitude error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_peak: float Comma-separated list of values, one per measured segment Range: -100 % to 100 % (AVERage: 0 % to 100 %, SDEViation: 0 % to 50 %) , Unit: %

7.3.9.3.2.8 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.average.
↪calculate()
```

Return magnitude error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_peak: float Comma-separated list of values, one per measured segment Range: -100 % to 100 % (AVERage: 0 % to 100 %, SDEViation: 0 % to 50 %) , Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:AVERage
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.average.
↪fetch()
```

Return magnitude error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_peak: float Comma-separated list of values, one per measured segment Range: -100 % to 100 % (AVERage: 0 % to 100 %, SDEViation: 0 % to 50 %) , Unit: %

7.3.9.3.2.9 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.maximum.
↪calculate()
```

Return magnitude error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_peak: float Comma-separated list of values, one per measured segment Range: -100 % to 100 % (AVERage: 0 % to 100 %, SDEViation: 0 % to 50 %) , Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.maximum.
↪fetch()
```

Return magnitude error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_peak: float Comma-separated list of values, one per measured segment Range: -100 % to 100 % (AVERage: 0 % to 100 %, SDEViation: 0 % to 50 %) , Unit: %

7.3.9.3.2.10 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.merror.peak.standardDev.
↪fetch()
```

Return magnitude error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return mag_error_peak: float Comma-separated list of values, one per measured segment Range: -100 % to 100 % (AVERage: 0 % to 100 %, SDEViation: 0 % to 50 %) , Unit: %

7.3.9.3.2.11 Percentile

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PERCentile
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PERCentile
```

class Percentile

Percentile commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PERCentile
value: List[float] = driver.multiEval.listPy.modulation.merror.percentile.
↪calculate()
```

Return magnitude error 95th percentile values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return magnitude_error: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PERCentile
value: List[float] = driver.multiEval.listPy.modulation.merror.percentile.
↪fetch()
```

Return magnitude error 95th percentile values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return magnitude_error: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.3.3 Perror

class Perror

Perror commands group definition. 16 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.clone()
```

Subgroups

7.3.9.3.3.1 Rms

class Rms

Rms commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.rms.clone()
```

Subgroups

7.3.9.3.3.2 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.current.
↪calculate()
```

Return phase error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_rms: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.current.
↪fetch()
```

Return phase error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_rms: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

7.3.9.3.3.3 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.average.
↪calculate()
```

Return phase error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_rms: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.average.
↪fetch()
```

Return phase error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_rms: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

7.3.9.3.3.4 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.maximum.
↪calculate()
```

Return phase error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_rms: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.maximum.
↪fetch()
```

Return phase error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_rms: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

7.3.9.3.3.5 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.rms.standardDev.
↪fetch()
```

Return phase error RMS values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_rms: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

7.3.9.3.3.6 Peak

class Peak

Peak commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.perror.peak.clone()
```

Subgroups

7.3.9.3.3.7 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.current.
↪calculate()
```

Return phase error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_peak: float Comma-separated list of values, one per measured segment Range: -180 deg to 180 deg (AVERage: 0 deg to 180 deg, SDEViation: 0 deg to 90 deg) , Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:CURRent
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.current.
↪fetch()
```

Return phase error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_peak: float Comma-separated list of values, one per measured segment Range: -180 deg to 180 deg (AVERage: 0 deg to 180 deg, SDEViation: 0 deg to 90 deg) , Unit: deg

7.3.9.3.3.8 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.average.
↪calculate()
```

Return phase error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_peak: float Comma-separated list of values, one per measured segment Range: -180 deg to 180 deg (AVERage: 0 deg to 180 deg, SDEViation: 0 deg to 90 deg) , Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:AVERage
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.average.
↪fetch()
```

Return phase error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_peak: float Comma-separated list of values, one per measured segment Range: -180 deg to 180 deg (AVERage: 0 deg to 180 deg, SDEViation: 0 deg to 90 deg) , Unit: deg

7.3.9.3.3.9 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.maximum.
↪calculate()
```

Return phase error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_peak: float Comma-separated list of values, one per measured segment Range: -180 deg to 180 deg (AVERage: 0 deg to 180 deg, SDEViation: 0 deg to 90 deg) , Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.maximum.
↪fetch()
```

Return phase error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_peak: float Comma-separated list of values, one per measured segment Range: -180 deg to 180 deg (AVERage: 0 deg to 180 deg, SDEViation: 0 deg to 90 deg) , Unit: deg

7.3.9.3.3.10 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.perror.peak.standardDev.
↪fetch()
```

Return phase error peak values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error_peak: float Comma-separated list of values, one per measured segment Range: -180 deg to 180 deg (AVERage: 0 deg to 180 deg, SDEViation: 0 deg to 90 deg) , Unit: deg

7.3.9.3.3.11 Percentile

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PERCentile
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PERCentile
```

class Percentile

Percentile commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PERCentile
value: List[float] = driver.multiEval.listPy.modulation.perror.percentile.
↪calculate()
```

Return phase error 95th percentile values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error: float Comma-separated list of values, one per measured segment Range: 0 deg to 180 deg, Unit: deg

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PERCentile
value: List[float] = driver.multiEval.listPy.modulation.perror.percentile.
↪fetch()
```

Return phase error 95th percentile values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return phase_error: float Comma-separated list of values, one per measured segment
Range: 0 deg to 180 deg, Unit: deg

7.3.9.3.4 IqOffset

class IqOffset

IqOffset commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.iqOffset.clone()
```

Subgroups

7.3.9.3.4.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:CURRent
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.current.
↪calculate()
```

Return I/Q origin offset results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_offset: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:CURRent
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.current.fetch()
```

Return I/Q origin offset results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_offset: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.4.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:AVERage
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.average.
↪calculate()
```

Return I/Q origin offset results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_offset: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:AVERage
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.average.fetch()
```

Return I/Q origin offset results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_offset: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.4.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.maximum.
↪calculate()
```

Return I/Q origin offset results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_offset: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.maximum.fetch()
```

Return I/Q origin offset results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_offset: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.4.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.iqOffset.standardDev.
↪fetch()
```

Return I/Q origin offset results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_offset: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.5 IqImbalance

class IqImbalance

IqImbalance commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.iqImbalance.clone()
```

Subgroups

7.3.9.3.5.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQIMbalance:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQIMbalance:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQIMbalance:CURRent
value: List[float] = driver.multiEval.listPy.modulation.iqImbalance.current.
↪calculate()
```

Return I/Q imbalance results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_imbalance: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQIMbalance:CURRent
value: List[float] = driver.multiEval.listPy.modulation.iqImbalance.current.
↪fetch()
```

Return I/Q imbalance results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_imbalance: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.5.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQIMbalance:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQIMbalance:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQIMbalance:AVERage
value: List[float] = driver.multiEval.listPy.modulation.iqImbalance.average.
↪calculate()
```

Return I/Q imbalance results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_imbalance: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQIMbalance:AVERage
value: List[float] = driver.multiEval.listPy.modulation.iqImbalance.average.
↪fetch()
```

Return I/Q imbalance results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_imbalance: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.5.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQIMbalance:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQIMbalance:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQIMbalance:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.iqImbalance.maximum.
↪calculate()
```

Return I/Q imbalance results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_imbalance: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQIMbalance:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.iqImbalance.maximum.
↪fetch()
```

Return I/Q imbalance results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_imbalance: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.5.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:IQIMbalance:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQIMbalance:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.iqImbalance.standardDev.
↪fetch()
```

Return I/Q imbalance results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return iq_imbalance: float Comma-separated list of values, one per measured segment
Range: -100 dB to 0 dB, Unit: dB

7.3.9.3.6 FreqError

class FreqError

FreqError commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.freqError.clone()
```

Subgroups

7.3.9.3.6.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:CURRent
value: List[float] = driver.multiEval.listPy.modulation.freqError.current.
↪calculate()
```

Return carrier frequency error results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return frequency_error: float Comma-separated list of values, one per measured segment Range: -56000 Hz to 56000 Hz, Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:CURRent
value: List[float] = driver.multiEval.listPy.modulation.freqError.current.
↪fetch()
```

Return carrier frequency error results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return frequency_error: float Comma-separated list of values, one per measured segment Range: -56000 Hz to 56000 Hz, Unit: Hz

7.3.9.3.6.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:AVERage
value: List[float] = driver.multiEval.listPy.modulation.freqError.average.
↪calculate()
```

Return carrier frequency error results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return frequency_error: float Comma-separated list of values, one per measured segment Range: -56000 Hz to 56000 Hz, Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:AVERage
value: List[float] = driver.multiEval.listPy.modulation.freqError.average.
↪fetch()
```

Return carrier frequency error results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return frequency_error: float Comma-separated list of values, one per measured segment Range: -56000 Hz to 56000 Hz, Unit: Hz

7.3.9.3.6.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.freqError.maximum.
↪calculate()
```

Return carrier frequency error results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return frequency_error: float Comma-separated list of values, one per measured segment Range: -56000 Hz to 56000 Hz, Unit: Hz

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.freqError.maximum.
↪fetch()
```

Return carrier frequency error results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return frequency_error: float Comma-separated list of values, one per measured segment Range: -56000 Hz to 56000 Hz, Unit: Hz

7.3.9.3.6.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.freqError.standardDev.
↪fetch()
```

Return carrier frequency error results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return frequency_error: float Comma-separated list of values, one per measured segment Range: -56000 Hz to 56000 Hz, Unit: Hz

7.3.9.3.7 Terror

class Terror

Terror commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.terror.clone()
```

Subgroups

7.3.9.3.7.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:CURRent
value: List[float] = driver.multiEval.listPy.modulation.terror.current.
↪calculate()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return timing_error: float Comma-separated list of values, one per measured segment
Range: -100 Sym to 100 Sym, Unit: Sym

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:CURRent
value: List[float] = driver.multiEval.listPy.modulation.terror.current.fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return timing_error: float Comma-separated list of values, one per measured segment
Range: -100 Sym to 100 Sym, Unit: Sym

7.3.9.3.7.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:AVERage
value: List[float] = driver.multiEval.listPy.modulation.terror.average.
↪calculate()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return timing_error: float Comma-separated list of values, one per measured segment
Range: -100 Sym to 100 Sym, Unit: Sym

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:AVERage
value: List[float] = driver.multiEval.listPy.modulation.terror.average.fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return timing_error: float Comma-separated list of values, one per measured segment
Range: -100 Sym to 100 Sym, Unit: Sym

7.3.9.3.7.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.terror.maximum.
↪calculate()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return timing_error: float Comma-separated list of values, one per measured segment
Range: -100 Sym to 100 Sym, Unit: Sym

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.terror.maximum.fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return timing_error: float Comma-separated list of values, one per measured segment
Range: -100 Sym to 100 Sym, Unit: Sym

7.3.9.3.7.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.terror.standardDev.
↪fetch()
```

Return transmit time error values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return timing_error: float Comma-separated list of values, one per measured segment
Range: -100 Sym to 100 Sym, Unit: Sym

7.3.9.3.8 Bpower

class Bpower

Bpower commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.bpower.clone()
```

Subgroups

7.3.9.3.8.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:CURRent
value: List[float] = driver.multiEval.listPy.modulation.bpower.current.
↪calculate()
```

Return burst power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:CURRent
value: List[float] = driver.multiEval.listPy.modulation.bpower.current.fetch()
```

Return burst power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.3.8.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:AVERage
value: List[float] = driver.multiEval.listPy.modulation.bpower.average.
↪calculate()
```

Return burst power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:AVERage
value: List[float] = driver.multiEval.listPy.modulation.bpower.average.fetch()
```

Return burst power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.3.8.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.bpower.maximum.
↪calculate()
```

Return burst power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.bpower.maximum.fetch()
```

Return burst power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.3.8.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:BPOWer:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:BPOWer:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.bpower.standardDev.
↪fetch()
```

Return burst power values for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return burst_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.3.9 ApDelay

class ApDelay

ApDelay commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.modulation.apDelay.clone()
```

Subgroups

7.3.9.3.9.1 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:APDelay:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:APDelay:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:APDelay:CURRent
value: List[float] = driver.multiEval.listPy.modulation.apDelay.current.
↪calculate()
```

Return AM-PM delay results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return am_pm_delay: float Comma-separated list of values, one per measured segment
Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:APDelay:CURRent
value: List[float] = driver.multiEval.listPy.modulation.apDelay.current.fetch()
```

Return AM-PM delay results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return am_pm_delay: float Comma-separated list of values, one per measured segment
Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

7.3.9.3.9.2 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:APDelay:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:APDelay:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:APDelay:AVERage
value: List[float] = driver.multiEval.listPy.modulation.apDelay.average.
↪calculate()
```

Return AM-PM delay results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return am_pm_delay: float Comma-separated list of values, one per measured segment
Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:APDelay:AVERage
value: List[float] = driver.multiEval.listPy.modulation.apDelay.average.fetch()
```

Return AM-PM delay results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return am_pm_delay: float Comma-separated list of values, one per measured segment
Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

7.3.9.3.9.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:APDelay:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:APDelay:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:APDelay:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.apDelay.maximum.
↪calculate()
```

Return AM-PM delay results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return am_pm_delay: float Comma-separated list of values, one per measured segment
Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:APDelay:MAXimum
value: List[float] = driver.multiEval.listPy.modulation.apDelay.maximum.fetch()
```

Return AM-PM delay results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return am_pm_delay: float Comma-separated list of values, one per measured segment
Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

7.3.9.3.9.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:APDelay:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:APDelay:SDEViation
value: List[float] = driver.multiEval.listPy.modulation.apDelay.standardDev.
↪fetch()
```

Return AM-PM delay results for all measured list mode segments. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return am_pm_delay: float Comma-separated list of values, one per measured segment
Range: -0.9225E-6 s to 0.9225E-6 s, Unit: s

7.3.9.3.10 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: List[float]: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: List[float]: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %) , Unit: %
- Phase_Error_Rms: List[float]: No parameter help available
- Phase_Error_Peak: List[float]: No parameter help available
- Iq_Offset: List[float]: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: List[float]: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: List[float]: float Average carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: List[float]: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: List[float]: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: List[float]: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: List[float]: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: List[float]: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %) , Unit: %
- Phase_Error_Rms: List[float]: No parameter help available
- Phase_Error_Peak: List[float]: No parameter help available
- Iq_Offset: List[float]: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: List[float]: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: List[float]: float Average carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: List[float]: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: List[float]: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: List[float]: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage
value: CalculateStruct = driver.multiEval.listPy.modulation.average.calculate()
```

Returns the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage
value: FetchStruct = driver.multiEval.listPy.modulation.average.fetch()
```

Returns the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.3.11 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: List[float]: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: List[float]: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %) , Unit: %
- Phase_Error_Rms: List[float]: No parameter help available

- Phase_Error_Peak: List[float]: No parameter help available
- Iq_Offset: List[float]: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: List[float]: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: List[float]: float Average carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: List[float]: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: List[float]: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: List[float]: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: List[float]: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: List[float]: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %) , Unit: %
- Phase_Error_Rms: List[float]: No parameter help available
- Phase_Error_Peak: List[float]: No parameter help available
- Iq_Offset: List[float]: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: List[float]: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: List[float]: float Average carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: List[float]: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: List[float]: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm

- Am_Pm_Delay: List[float]: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent
value: CalculateStruct = driver.multiEval.listPy.modulation.current.calculate()
```

Returns the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent
value: FetchStruct = driver.multiEval.listPy.modulation.current.fetch()
```

Returns the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.3.12 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.

- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: List[float]: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: List[float]: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %) , Unit: %
- Phase_Error_Rms: List[float]: No parameter help available
- Phase_Error_Peak: List[float]: No parameter help available
- Iq_Offset: List[float]: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: List[float]: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: List[float]: float Average carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: List[float]: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: List[float]: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: List[float]: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %

- Evm_Peak: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: List[float]: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: List[float]: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %) , Unit: %
- Phase_Error_Rms: List[float]: No parameter help available
- Phase_Error_Peak: List[float]: No parameter help available
- Iq_Offset: List[float]: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: List[float]: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: List[float]: float Average carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: List[float]: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: List[float]: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: List[float]: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MAXimum
value: CalculateStruct = driver.multiEval.listPy.modulation.maximum.calculate()
```

Returns the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:MAXimum
value: FetchStruct = driver.multiEval.listPy.modulation.maximum.fetch()
```

Returns the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.3.13 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 50 %, Unit: %
- Evm_Peak: List[float]: float Error vector magnitude RMS and peak value Range: 0 % to 50 %, Unit: %
- Mag_Error_Rms: List[float]: float Magnitude error RMS and peak value Range: 0 % to 50 %, Unit: %
- Mag_Error_Peak: List[float]: float Magnitude error RMS and peak value Range: 0 % to 50 %, Unit: %
- Phase_Error_Rms: List[float]: No parameter help available
- Phase_Error_Peak: List[float]: No parameter help available
- Iq_Offset: List[float]: float I/Q origin offset Range: 0 dB to 50 dB, Unit: dB
- Iq_Imbalance: List[float]: float I/Q imbalance Range: 0 dB to 50 dB, Unit: dB
- Frequency_Error: List[float]: float Carrier frequency error Range: 0 Hz to 56000 Hz, Unit: Hz
- Timing_Error: List[float]: float Transmit time error Range: 0 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: List[float]: float Burst power Range: 0 dB to 71 dB, Unit: dB
- Am_Pm_Delay: List[float]: float AM-PM delay (determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned) Range: 0 s to 0.9225E-6 s, Unit: s

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation
value: FetchStruct = driver.multiEval.listPy.modulation.standardDev.fetch()
```

Returns the standard deviation of the modulation results in list mode. The values listed below in curly brackets { } are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.3.14 Percentile

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERCentile
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERCentile
```

class Percentile

Percentile commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: GMSK | EPSK | ACCess | Q16 | OFF Detected burst type of the last measured burst GMSK: Normal burst, GMSK-modulated EPSK: Normal burst, 8PSK-modulated ACCess: Access burst Q16: Normal burst, 16-QAM-modulated OFF: Inactive slot
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm: List[enums.ResultStatus2]: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Magnitude_Error: List[enums.ResultStatus2]: float Magnitude error percentile Range: 0 % to 100 %, Unit: %
- Phase_Error: List[enums.ResultStatus2]: float Phase error percentile Range: 0 deg to 180 deg, Unit: deg

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.

- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: List[int]: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: List[enums.SlotInfo]: GMSK | EPSK | ACCess | Q16 | OFF Detected burst type of the last measured burst GMSK: Normal burst, GMSK-modulated EPSK: Normal burst, 8PSK-modulated ACCess: Access burst Q16: Normal burst, 16-QAM-modulated OFF: Inactive slot
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm: List[float]: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Magnitude_Error: List[float]: float Magnitude error percentile Range: 0 % to 100 %, Unit: %
- Phase_Error: List[float]: float Phase error percentile Range: 0 deg to 180 deg, Unit: deg

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERCentile
value: CalculateStruct = driver.multiEval.listPy.modulation.percentile.
↪calculate()
```

Returns the 95th percentile of the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:MODulation:PERCentile
value: FetchStruct = driver.multiEval.listPy.modulation.percentile.fetch()
```

Returns the 95th percentile of the modulation results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.4 Smodulation

class Smodulation
Smodulation commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.smodulation.clone()
```

Subgroups

7.3.9.4.1 Cpower

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:CPOWer
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:CPOWer
```

class Cpower
Cpower commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:CPOWer
value: List[float] = driver.multiEval.listPy.smodulation.cpower.calculate()
```

Return carrier output power results for all measured list mode segments, for spectrum due to modulation or spectrum due to switching measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return carrier_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:CPOWer
value: List[float] = driver.multiEval.listPy.smodulation.cpower.fetch()
```

Return carrier output power results for all measured list mode segments, for spectrum due to modulation or spectrum due to switching measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return carrier_power: float Comma-separated list of values, one per measured segment
Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.4.2 Poffset<FreqOffset>

RepCap Settings

```
# Range: Nr1 .. Nr41
rc = driver.multiEval.listPy.smodulation.poffset.repcap_freqOffset_get()
driver.multiEval.listPy.smodulation.poffset.repcap_freqOffset_set(repcap.FreqOffset.Nr1)
```

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:POFFset<FreqOffset>
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:POFFset<FreqOffset>
```

class Poffset

Poffset commands group definition. 2 total commands, 0 Sub-groups, 2 group commands Repeated Capability: FreqOffset, default value after init: FreqOffset.Nr1

calculate(*freqOffset=<FreqOffset.Default: -1>*) → List[RsCmwGsmMeas.enums.ResultStatus2]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:POFFset
↪<nr>
value: List[enums.ResultStatus2] = driver.multiEval.listPy.smodulation.poffset.
↪calculate(freqOffset = repcap.FreqOffset.Default)
```

Return the burst power at the carrier frequency minus/plus a selected frequency offset, for all measured list mode segments of the spectrum due to modulation measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param freqOffset optional repeated capability selector. Default value: Nr1 (settable in the interface 'Poffset')

return power: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

fetch(*freqOffset=<FreqOffset.Default: -1>*) → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SMODulation:POFFset<nr>
value: List[float] = driver.multiEval.listPy.smodulation.poffset.
↪fetch(freqOffset = repcap.FreqOffset.Default)
```

Return the burst power at the carrier frequency minus/plus a selected frequency offset, for all measured list mode segments of the spectrum due to modulation measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param freqOffset optional repeated capability selector. Default value: Nr1 (settable in the interface 'Poffset')

return power: float Comma-separated list of values, one per measured segment Range: -100 dB to 100 dB, Unit: dB

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.smodulation.poffset.clone()
```

7.3.9.5 Sswitching

class Sswitching

Sswitching commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.sswitching.clone()
```

Subgroups

7.3.9.5.1 Cpower

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:CPOWer
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:CPOWer
```

class Cpower

Cpower commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:CPOWer
value: List[float] = driver.multiEval.listPy.sswitching.cpower.calculate()
```

Return carrier output power results for all measured list mode segments, for spectrum due to modulation or spectrum due to switching measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return carrier_power: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:CPOWer
value: List[float] = driver.multiEval.listPy.sswitching.cpower.fetch()
```

Return carrier output power results for all measured list mode segments, for spectrum due to modulation or spectrum due to switching measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return carrier_power: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

7.3.9.5.2 Poffset<FreqOffset>

RepCap Settings

```
# Range: Nr1 .. Nr41
rc = driver.multiEval.listPy.sswitching.poffset.repcap_freqOffset_get()
driver.multiEval.listPy.sswitching.poffset.repcap_freqOffset_set(repcap.FreqOffset.Nr1)
```

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:POFFset<FreqOffset>
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:POFFset<FreqOffset>
```

class Poffset

Poffset commands group definition. 2 total commands, 0 Sub-groups, 2 group commands Repeated Capability: FreqOffset, default value after init: FreqOffset.Nr1

calculate(*freqOffset=<FreqOffset.Default: -1>*) → List[RsCmwGsmMeas.enums.ResultStatus2]

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:POFFset
↪<nr>
value: List[enums.ResultStatus2] = driver.multiEval.listPy.sswitching.poffset.
↪calculate(freqOffset = repcap.FreqOffset.Default)
```

Return the burst power at the carrier frequency minus/plus a selected frequency offset, for all measured list mode segments of the spectrum due to switching measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param freqOffset optional repeated capability selector. Default value: Nr1 (settable in the interface 'Poffset')

return power: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

fetch(*freqOffset=<FreqOffset.Default: -1>*) → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SSWitching:POFFset<nr>
value: List[float] = driver.multiEval.listPy.sswitching.poffset.
↪fetch(freqOffset = repcap.FreqOffset.Default)
```

Return the burst power at the carrier frequency minus/plus a selected frequency offset, for all measured list mode segments of the spectrum due to switching measurement. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

param freqOffset optional repeated capability selector. Default value: Nr1 (settable in the interface 'Poffset')

return power: float Comma-separated list of values, one per measured segment Range: -100 dBm to 55 dBm, Unit: dBm

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.sswitching.poffset.clone()
```

7.3.9.6 Ber

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER
```

class Ber

Ber commands group definition. 4 total commands, 3 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: List[int]: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statistic_Expire: List[int]: No parameter help available
- Slot_Info: List[enums.SlotInfo]: No parameter help available
- Slot_Statistic: List[bool]: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Ber: List[float]: float % bit error rate Range: 0 % to 100 %, Unit: %
- Ber_Absolute: List[int or bool]: decimal Total number of detected bit errors The BER measurement evaluates: 114 data bits per GMSK-modulated normal burst 306 data bits per 8PSK-modulated burst. Range: 0 to no. of measured bits
- Ber_Count: List[int or bool]: decimal Total number of measured bursts Range: 0 to StatisticCount For StatisticCount, see [CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:BER CMDLINK]

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER
value: FetchStruct = driver.multiEval.listPy.ber.fetch()
```

Returns the BER results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange.

return structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.ber.clone()
```

Subgroups

7.3.9.6.1 Ber

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER:BER
```

class Ber

Ber commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER:BER
value: List[float] = driver.multiEval.listPy.ber.ber.fetch()
```

Returns the bit error rate for each measured list mode segment.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return ber: float Comma-separated list of values, one per measured segment Range: 0 % to 100 %, Unit: %

7.3.9.6.2 Absolute

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER:ABSolute
```

class Absolute

Absolute commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER:ABSolute
value: List[int or bool] = driver.multiEval.listPy.ber.absolute.fetch()
```

Returns the total number of detected bit errors for each measured list mode segment.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return ber_absolute: Comma-separated list of values, one per measured segment Range: 0 to no. of measured bits

7.3.9.6.3 Count

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER:COUNt
```

class Count

Count commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:BER:COUNt
value: List[int or bool] = driver.multiEval.listPy.ber.count.fetch()
```

Returns the number of measured bursts for each list mode segment.

Use RsCmwGsmMeas.reliability.last_value to read the updated reliability indicator.

return ber_count: Comma-separated list of values, one per measured segment Range: 0 to StatisticCount

7.3.9.7 Overview

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:OVERview
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:OVERview
```

class Overview

Overview commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Segm_Reliability: List[int]: No parameter help available
- Out_Of_Tol: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Avg_Burst_Power: List[float]: No parameter help available
- Evm_Rms_Avg: List[float]: No parameter help available
- Evm_Peak_Max: List[float]: No parameter help available
- Evm_95_Perc: List[float]: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Phase_Error_Rms_Avg: List[float]: No parameter help available

- Phase_Error_Peak_Max: List[float]: No parameter help available
- Iq_Offset_Avg: List[float]: No parameter help available
- Frequency_Error_Avg: List[float]: No parameter help available
- Spec_Mod_Offs_N_5: List[enums.ResultStatus2]: No parameter help available
- Spec_Mod_Offs_N_4: List[enums.ResultStatus2]: No parameter help available
- Spec_Mod_Carrier: List[float]: No parameter help available
- Spec_Mod_Offs_P_4: List[enums.ResultStatus2]: No parameter help available
- Spec_Mod_Offs_P_5: List[enums.ResultStatus2]: No parameter help available
- Spec_Switch_Offs_N_2: List[enums.ResultStatus2]: No parameter help available
- Spec_Switch_Offs_N_1: List[enums.ResultStatus2]: No parameter help available
- Spec_Switch_Carrier: List[enums.ResultStatus2]: No parameter help available
- Spec_Switch_Offs_P_1: List[enums.ResultStatus2]: No parameter help available
- Spec_Switch_Offs_P_2: List[enums.ResultStatus2]: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator'
- Segm_Reliability: List[int]: No parameter help available
- Out_Of_Tol: List[int]: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Avg_Burst_Power: List[float]: No parameter help available
- Evm_Rms_Avg: List[float]: No parameter help available
- Evm_Peak_Max: List[float]: No parameter help available
- Evm_95_Perc: List[float]: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Phase_Error_Rms_Avg: List[float]: No parameter help available
- Phase_Error_Peak_Max: List[float]: No parameter help available
- Iq_Offset_Avg: List[float]: No parameter help available
- Frequency_Error_Avg: List[float]: No parameter help available
- Spec_Mod_Offs_N_5: List[float]: No parameter help available
- Spec_Mod_Offs_N_4: List[float]: No parameter help available
- Spec_Mod_Carrier: List[float]: No parameter help available
- Spec_Mod_Offs_P_4: List[float]: No parameter help available
- Spec_Mod_Offs_P_5: List[float]: No parameter help available
- Spec_Switch_Offs_N_2: List[float]: No parameter help available
- Spec_Switch_Offs_N_1: List[float]: No parameter help available
- Spec_Switch_Carrier: List[float]: No parameter help available
- Spec_Switch_Offs_P_1: List[float]: No parameter help available
- Spec_Switch_Offs_P_2: List[float]: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:OVERview
value: CalculateStruct = driver.multiEval.listPy.overview.calculate()
```

Returns all single results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:OVERview
value: FetchStruct = driver.multiEval.listPy.overview.fetch()
```

Returns all single results in list mode. The values listed below in curly brackets {} are returned for each measured segment: {...}seg 1, {...}seg 2, ..., {...}seg n. The position of measured segments within the range of configured segments and their number n is determined by method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange. The values described below are returned by FETCh commands. CALCulate commands return limit check results instead, one value for each result listed below.

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8 Segment

RepCap Settings

```
# Range: Nr1 .. Nr512
rc = driver.multiEval.listPy.segment.repcap_segment_get()
driver.multiEval.listPy.segment.repcap_segment_set(repcap.Segment.Nr1)
```

class Segment

Segment commands group definition. 22 total commands, 5 Sub-groups, 0 group commands Repeated Capability: Segment, default value after init: Segment.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.clone()
```

Subgroups

7.3.9.8.1 PowerVsTime

class PowerVsTime
PowerVsTime commands group definition. 8 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.powerVsTime.clone()
```

Subgroups

7.3.9.8.1.1 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:AVERage
```

class Average
Average commands group definition. 3 total commands, 1 Sub-groups, 2 group commands

class CalculateStruct
Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: float: No parameter help available

class FetchStruct
Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)

- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: float: No parameter help available

calculate(*segment=<Segment.Default: -1>*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.powerVsTime.average.
↪calculate(segment = repcap.Segment.Default)
```

Returns power vs. time results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.powerVsTime.average.
↪fetch(segment = repcap.Segment.Default)
```

Returns power vs. time results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.powerVsTime.average.clone()
```

Subgroups

7.3.9.8.1.2 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:AVERage:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch(*segment*=<*Segment.Default: -1*>) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:AVERage:SVECtor
value: FetchStruct = driver.multiEval.listPy.segment.powerVsTime.average.
↪svector.fetch(segment = repcap.Segment.Default)
```

Returns special burst power results for segment <no> in list mode.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.1.3 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:CURRent
```

class Current

Current commands group definition. 3 total commands, 1 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: float: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Average_Burst_Pow: float: No parameter help available

calculate(*segment=<Segment.Default: -1>*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.powerVsTime.current.
↪calculate(segment = repcap.Segment.Default)
```

Returns power vs. time results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.powerVsTime.current.
↪fetch(segment = repcap.Segment.Default)
```

Returns power vs. time results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.powerVsTime.current.clone()
```

Subgroups

7.3.9.8.1.4 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:CURRent:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)

- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:CURRent:SVECtor
value: FetchStruct = driver.multiEval.listPy.segment.powerVsTime.current.
↪svector.fetch(segment = repcap.Segment.Default)
```

Returns special burst power results for segment <no> in list mode.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.1.5 Minimum

class Minimum

Minimum commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.powerVsTime.minimum.clone()
```

Subgroups

7.3.9.8.1.6 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:MINimum:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch(*segment*=<*Segment.Default: -1*>) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:MINimum:SVECtor
value: FetchStruct = driver.multiEval.listPy.segment.powerVsTime.minimum.
↪svector.fetch(segment = repcap.Segment.Default)
```

Returns special burst power results for segment <no> in list mode.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.1.7 Maximum

class Maximum

Maximum commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.powerVsTime.maximum.clone()
```

Subgroups

7.3.9.8.1.8 Svector

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:PVTime:MAXimum:SVECtor
```

class Svector

Svector commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Usefull_Part_Min: float: No parameter help available
- Usefull_Part_Max: float: No parameter help available
- Subvector: List[float]: No parameter help available

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PVTime:MAXimum:SVECtor
value: FetchStruct = driver.multiEval.listPy.segment.powerVsTime.maximum.
↪svector.fetch(segment = repcap.Segment.Default)
```

Returns special burst power results for segment <no> in list mode.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.2 Modulation

class Modulation

Modulation commands group definition. 9 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.multiEval.listPy.segment.modulation.clone()
```

Subgroups

7.3.9.8.2.1 Average

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:AVERage
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:AVERage
```

class Average

Average commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %, SDEViation: 0 % to 50 %) , Unit: %
- Phase_Error_Rms: float: No parameter help available
- Phase_Error_Peak: float: No parameter help available
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB

- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay, determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %, SDEViation: 0 % to 50 %) , Unit: %
- Phase_Error_Rms: float: No parameter help available
- Phase_Error_Peak: float: No parameter help available
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay, determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

calculate(*segment=<Segment.Default: -1>*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage
value: CalculateStruct = driver.multiEval.listPy.segment.modulation.average.
↪calculate(segment = repcap.Segment.Default)
```

Returns the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by

CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage
value: FetchStruct = driver.multiEval.listPy.segment.modulation.average.
↪fetch(segment = repcap.Segment.Default)
```

Returns the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.2.2 Current

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:CURRent
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %

- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %, SDEViation: 0 % to 50 %) , Unit: %
- Phase_Error_Rms: float: No parameter help available
- Phase_Error_Peak: float: No parameter help available
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay, determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %, SDEViation: 0 % to 50 %) , Unit: %
- Phase_Error_Rms: float: No parameter help available
- Phase_Error_Peak: float: No parameter help available
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay, determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

calculate(*segment=<Segment.Default: -1>*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:CURRent
value: CalculateStruct = driver.multiEval.listPy.segment.modulation.current.
↪calculate(segment = repcap.Segment.Default)
```

Returns the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:CURRent
value: FetchStruct = driver.multiEval.listPy.segment.modulation.current.
↪fetch(segment = repcap.Segment.Default)
```

Returns the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.2.3 Maximum

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:MAXimum
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:MAXimum
```

class Maximum

Maximum commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)

- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %, SDEViation: 0 % to 50 %) , Unit: %
- Phase_Error_Rms: float: No parameter help available
- Phase_Error_Peak: float: No parameter help available
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay, determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %, SDEViation: 0 % to 50 %) , Unit: %
- Phase_Error_Rms: float: No parameter help available
- Phase_Error_Peak: float: No parameter help available
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB

- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay, determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

calculate(*segment=<Segment.Default: -1>*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:MAXimum
value: CalculateStruct = driver.multiEval.listPy.segment.modulation.maximum.
↪calculate(segment = repcap.Segment.Default)
```

Returns the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:MAXimum
value: FetchStruct = driver.multiEval.listPy.segment.modulation.maximum.
↪fetch(segment = repcap.Segment.Default)
```

Returns the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.2.4 StandardDev

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:SDEViation
```

class StandardDev

StandardDev commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm_Rms: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Evm_Peak: float: float Error vector magnitude RMS and peak value Range: 0 % to 100 %, Unit: %
- Mag_Error_Rms: float: float Magnitude error RMS value Range: 0 % to 100 %, Unit: %
- Mag_Error_Peak: float: float Magnitude error peak value Range: -100 % to 100 % (AVERage: 0% to 100 %, SDEViation: 0 % to 50 %) , Unit: %
- Phase_Error_Rms: float: No parameter help available
- Phase_Error_Peak: float: No parameter help available
- Iq_Offset: float: float I/Q origin offset Range: -100 dB to 0 dB, Unit: dB
- Iq_Imbalance: float: float I/Q imbalance Range: -100 dB to 0 dB, Unit: dB
- Frequency_Error: float: float Carrier frequency error Range: -56000 Hz to 56000 Hz, Unit: Hz
- Timing_Error: float: float Transmit time error Range: -100 Symbol to 100 Symbol, Unit: Symbol
- Burst_Power: float: float Burst power Range: -100 dBm to 55 dBm, Unit: dBm
- Am_Pm_Delay: float: float AM-PM delay, determined for 8PSK and 16-QAM modulation only - for GMSK zeros are returned Range: -0.9225E-6 s to 0.9225E-6 s (a quarter of a symbol period) , Unit: s

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:SDEViation
value: FetchStruct = driver.multiEval.listPy.segment.modulation.standardDev.
↪fetch(segment = repcap.Segment.Default)
```

Returns the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.2.5 Percentile

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:MODulation:PERCentile
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>
↪:MODulation:PERCentile
```

class Percentile

Percentile commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Segment_Reliability: int: No parameter help available
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm: enums.ResultStatus2: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Magnitude_Error: enums.ResultStatus2: float Magnitude error percentile Range: 0 % to 100 %, Unit: %
- Phase_Error: enums.ResultStatus2: float Phase error percentile Range: 0 deg to 180 deg, Unit: deg

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Segment_Reliability: int: No parameter help available
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Evm: float: float Error vector magnitude percentile Range: 0 % to 100 %, Unit: %
- Magnitude_Error: float: float Magnitude error percentile Range: 0 % to 100 %, Unit: %
- Phase_Error: float: float Phase error percentile Range: 0 deg to 180 deg, Unit: deg

calculate(*segment=<Segment.Default: -1>*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:PERCentile
value: CalculateStruct = driver.multiEval.listPy.segment.modulation.percentile.
↪calculate(segment = repcap.Segment.Default)
```

Returns the 95th percentile of the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:PERCentile
value: FetchStruct = driver.multiEval.listPy.segment.modulation.percentile.
↪fetch(segment = repcap.Segment.Default)
```

Returns the 95th percentile of the modulation results for segment <no> in list mode. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.3 Smodulation

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:SMODulation
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:SMODulation
```

class Smodulation

Smodulation commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available

- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Carrier_Power: enums.ResultStatus2: float Measured carrier output power (reference power) Range: -100 dBm to 55 dBm, Unit: dBm
- Power: List[enums.ResultStatus2]: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Carrier_Power: float: float Measured carrier output power (reference power) Range: -100 dBm to 55 dBm, Unit: dBm
- Power: List[float]: No parameter help available

calculate(*segment=<Segment.Default: -1>*) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SMODulation
value: CalculateStruct = driver.multiEval.listPy.segment.smodulation.
↪calculate(segment = repcap.Segment.Default)
```

Returns the spectrum due to modulation results for segment <no> in list mode. The result is averaged over the statistical length. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment=<Segment.Default: -1>*) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SMODulation
value: FetchStruct = driver.multiEval.listPy.segment.smodulation.fetch(segment␣
↪= repcap.Segment.Default)
```

Returns the spectrum due to modulation results for segment <no> in list mode. The result is averaged over the statistical length. The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.4 Sswitching

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:SSWitching
CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:SSWitching
```

class Sswitching

Sswitching commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Carrier_Power: enums.ResultStatus2: float Measured carrier output power (reference power) Range: -100 dBm to 55 dBm, Unit: dBm
- Power: List[enums.ResultStatus2]: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statist_Expired: int: decimal Number of measured steps Range: 0 to Statistical Length (integer value)
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range

- Out_Of_Tolerance: int: decimal Percentage of measured bursts with failed limit check Range: 0 % to 100 %, Unit: %
- Carrier_Power: float: float Measured carrier output power (reference power) Range: -100 dBm to 55 dBm, Unit: dBm
- Power: List[float]: No parameter help available

calculate(*segment*=<*Segment.Default: -1*>) → CalculateStruct

```
# SCPI: CALCulate:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SSWitching
value: CalculateStruct = driver.multiEval.listPy.segment.sswitching.
↪calculate(segment = repcap.Segment.Default)
```

Returns the spectrum due to switching results for segment <no> in list mode. The result corresponds to the maximum over the statistical length (peak hold mode) . The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for CalculateStruct structure arguments.

fetch(*segment*=<*Segment.Default: -1*>) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SSWitching
value: FetchStruct = driver.multiEval.listPy.segment.sswitching.fetch(segment =␣
↪repcap.Segment.Default)
```

Returns the spectrum due to switching results for segment <no> in list mode. The result corresponds to the maximum over the statistical length (peak hold mode) . The values described below are returned by FETCh commands. The first six values ('Reliability' to 'Out of Tolerance' result) are also returned by CALCulate commands. The remaining values returned by CALCulate commands are limit check results, one value for each result listed below.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.3.9.8.5 Ber

SCPI Commands

```
FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<Segment>:BER
```

class Ber

Ber commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: decimal 'Reliability Indicator' In list mode, a zero reliability indicator indicates that the results in all measured segments are valid. A non-zero value indicates that an error occurred in at least one of the measured segments.
- Seg_Reliability: int: decimal Reliability indicator for the segment. The meaning of the returned values is the same as for the common reliability indicator, see previous parameter.
- Statistic_Expire: int: No parameter help available
- Slot_Info: enums.SlotInfo: No parameter help available
- Slot_Statistic: bool: ON | OFF ON: Averaging over different burst type OFF: Uniform burst type in the averaging range
- Ber: float: float % bit error rate Range: 0 % to 100 %, Unit: %
- Ber_Absolute: int or bool: decimal Total number of detected bit errors The BER measurement evaluates: 114 data bits per GMSK-modulated normal burst 306 data bits per 8PSK-modulated burst. Range: 0 to no. of measured bits
- Ber_Count: int or bool: decimal Total number of measured bursts Range: 0 to StatisticCount For StatisticCount, see [CMDLINK: CONFigure:GSM:MEASi:MEValuation:SCOunt:BER CMDLINK]

fetch(*segment*=<*Segment.Default: -1*>) → FetchStruct

```
# SCPI: FETCh:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:BER
value: FetchStruct = driver.multiEval.listPy.segment.ber.fetch(segment = repcap.
↪Segment.Default)
```

Returns the BER results for segment <no> in list mode.

param segment optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return structure: for return value, see the help for FetchStruct structure arguments.

7.4 Trigger

class Trigger

Trigger commands group definition. 7 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.clone()
```

Subgroups

7.4.1 MultiEval

SCPI Commands

```
TRIGger:GSM:MEASurement<Instance>:MEValuation:SOURce
TRIGger:GSM:MEASurement<Instance>:MEValuation:THReshold
TRIGger:GSM:MEASurement<Instance>:MEValuation:SLOPe
TRIGger:GSM:MEASurement<Instance>:MEValuation:TOUT
TRIGger:GSM:MEASurement<Instance>:MEValuation:MGAP
```

class MultiEval

MultiEval commands group definition. 7 total commands, 2 Sub-groups, 5 group commands

get_mgap() → int

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:MGAP
value: int = driver.trigger.multiEval.get_mgap()
```

Sets a minimum time during which the IF signal must be below the trigger threshold before the trigger is armed so that an IF power trigger event can be generated.

return min_trigger_gap: integer Range: 1 slot to 7 slots, Unit: slot

get_slope() → RsCmwGsmMeas.enums.SignalSlope

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:SLOPe
value: enums.SignalSlope = driver.trigger.multiEval.get_slope()
```

Qualifies whether the trigger event is generated at the rising or at the falling edge of the trigger pulse (valid for external and power trigger sources) .

return slope: REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

get_source() → str

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.multiEval.get_source()
```

Selects the source of the trigger events. Some values are always available. They are listed below. Depending on the installed options, additional values are available. You can query a list of all supported values via TRIGger:... :CATalog:SOURce?.

return source: string 'Power': Power trigger (received RF power) 'Acquisition': Frame trigger according to defined burst pattern 'Free Run': Free run (untriggered)

get_threshold() → float

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:THReshold
value: float or bool = driver.trigger.multiEval.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return threshold: numeric | ON | OFF Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation) Additional parameters: OFF | ON (disables | enables the threshold)

get_timeout() → float

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:TOUT
value: float or bool = driver.trigger.multiEval.get_timeout()
```

Selects the maximum time that the measurement waits for a trigger event before it stops in remote control mode or indicates a trigger timeout in manual operation mode. This setting has no influence on 'Free Run' measurements.

return trigger_timeout: numeric | ON | OFF Range: 0.01 s to 167.77215E+3 s, Unit: s Additional parameters: OFF | ON (disables timeout | enables timeout using the previous/default values)

set_mgap(*min_trigger_gap: int*) → None

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:MGAP
driver.trigger.multiEval.set_mgap(min_trigger_gap = 1)
```

Sets a minimum time during which the IF signal must be below the trigger threshold before the trigger is armed so that an IF power trigger event can be generated.

param min_trigger_gap integer Range: 1 slot to 7 slots, Unit: slot

set_slope(*slope: RsCmwGsmMeas.enums.SignalSlope*) → None

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:SLOPe
driver.trigger.multiEval.set_slope(slope = enums.SignalSlope.FEDGe)
```

Qualifies whether the trigger event is generated at the rising or at the falling edge of the trigger pulse (valid for external and power trigger sources) .

param slope REDGe | FEDGe REDGe: Rising edge FEDGe: Falling edge

set_source(*source: str*) → None

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.multiEval.set_source(source = '1')
```

Selects the source of the trigger events. Some values are always available. They are listed below. Depending on the installed options, additional values are available. You can query a list of all supported values via TRIGger:... :CATalog:SOURce?.

param source string 'Power': Power trigger (received RF power) 'Acquisition': Frame trigger according to defined burst pattern 'Free Run': Free run (untriggered)

set_threshold(*threshold: float*) → None

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:THReshold
driver.trigger.multiEval.set_threshold(threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param threshold numeric | ON | OFF Range: -50 dB to 0 dB, Unit: dB (full scale, i.e. relative to reference level minus external attenuation) Additional parameters: OFF | ON (disables | enables the threshold)

set_timeout(*trigger_timeout: float*) → None

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:TOUT
driver.trigger.multiEval.set_timeout(trigger_timeout = 1.0)
```

Selects the maximum time that the measurement waits for a trigger event before it stops in remote control mode or indicates a trigger timeout in manual operation mode. This setting has no influence on 'Free Run' measurements.

param trigger_timeout numeric | ON | OFF Range: 0.01 s to 167.77215E+3 s, Unit: s Additional parameters: OFF | ON (disables timeout | enables timeout using the previous/default values)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.multiEval.clone()
```

Subgroups

7.4.1.1 Catalog

SCPI Commands

```
TRIGger:GSM:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class Catalog

Catalog commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.multiEval.catalog.get_source()
```

Lists all trigger source values that can be set using method RsCmwGsmMeas.Trigger.MultiEval.source.

return source_list: string Comma-separated list of all supported values. Each value is represented as a string.

7.4.1.2 ListPy

SCPI Commands

```
TRIGger:GSM:MEASurement<Instance>:MEValuation:LIST:MODE
```

class ListPy

ListPy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_mode() → RsCmwGsmMeas.enums.ListMode

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:LIST:MODE
value: enums.ListMode = driver.trigger.multiEval.listPy.get_mode()
```

Specifies whether a trigger event initiates a measurement of the entire measurement interval (comprising the number of segments defined via method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange) or a measurement of single segment.

return mode: ONCE | SEGMent ONCE: A trigger event is only required to start the measurement. The entire range of segments to be measured is captured without additional trigger event. The retrigger flags of the segments are ignored. SEGMent: The retrigger flag of each segment is evaluated. It defines whether the measurement waits for a trigger event before capturing the segment, or not.

set_mode(*mode: RsCmwGsmMeas.enums.ListMode*) → None

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:LIST:MODE
driver.trigger.multiEval.listPy.set_mode(mode = enums.ListMode.ONCE)
```

Specifies whether a trigger event initiates a measurement of the entire measurement interval (comprising the number of segments defined via method RsCmwGsmMeas.Configure.MultiEval.ListPy.lrange) or a measurement of single segment.

param mode ONCE | SEGMent ONCE: A trigger event is only required to start the measurement. The entire range of segments to be measured is captured without additional trigger event. The retrigger flags of the segments are ignored. SEGMent: The retrigger flag of each segment is evaluated. It defines whether the measurement waits for a trigger event before capturing the segment, or not.

INDEX

116
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
114
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
125
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
124
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
128
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
127
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
106
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
83
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
83
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
83
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
83
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
83
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
100
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
98
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
96
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
95
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
90
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
88
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
93
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST,
92
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:CMWS
102
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:IIFRa
101
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:LRANg
105
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:OSINd
104
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMe
83
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMe
82
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMe

81
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
144
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
150
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
148
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
147
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
145
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
146
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
143
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
141
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
139
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
138
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
132
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
131
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
136
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
134
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
152
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
151
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
155
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
154
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIMit:QAM
149
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST,
58
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:CMWS
69
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:IIFRa
58
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:LRANg
58
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:OSINd
58
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMe
66
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMe
68
CONFigure:GSM:MEASurement<Instance>:MEValuation:LIST:SEGMe

F

I

R

S

T