RsCMPX_LteMeas

Release 5.0.70.14

Rohde & Schwarz

Apr 18, 2024

CONTENTS:


5G
CMX/P

CHAPTER

ONE

REVISION HISTORY

1.1 RsCMPX_LteMeas

Rohde & Schwarz CMX/CMP LTE Measurement RsCMPX_LteMeas instrument driver.

Basic Hello-World code:

```
from RsCMPX_LteMeas import *

instr = RsCMPX_LteMeas('TCPIP::192.168.2.101::hislip0')
idn = instr.query('*IDN?')
print('Hello, I am: ' + idn)
```

Supported instruments: CMX500, CMP180, PVT360

The package is hosted here: https://pypi.org/project/RsCMPX-LteMeas/

Documentation: https://RsCMPX-LteMeas.readthedocs.io/

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

1.1.1 Version history

Latest release notes summary: Update for FW 5.0.70

Version 5.0.70

- Update for FW 5.0.70

Version 4.0.186

- Fixed Documentation

Version 4.0.185

- Update to FW 4.0.185

Version 4.0.140

- Update of RsCMPX_LteMeas to FW 4.0.140 from the complete FW package 7.10.0

Version 4.0.60

- Update of RsCMPX_LteMeas to FW 4.0.60

Version 4.0.10

- First released version

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

RsCMPX_LteMeas 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 example for RsCmpx-Base and RsCmpx-Gprf:

```
"""
# GitHub examples repository path: CMXP/Python/RsCmxp_xxx_ScpiPackages

Example on how to use the python RsCmx auto-generated instrument drivers for
RsCmpx_Base and RsCmpx_Gprf (Base and GPRF) in one script with shared VISA session.
"""

from RsCMPX_Base.RsCMPX_Base import RsCMPX_Base  # install from pypi.org
from RsCMPX_Base import enums as base_enums
from RsCMPX_Base import repcap as base_repcap

from RsCMPX_Gprf.RsCMPX_Gprf import RsCMPX_Gprf  # install from pypi.org
from RsCMPX_Gprf.CustomFiles.reliability import ReliabilityEventArgs
from RsCMPX_Gprf import enums as gprf_enums
from RsCMPX_Gprf import repcap as gprf_repcaps
```

(continues on next page)

(continued from previous page)

```
# CMX Base init
cmx_base = RsCMPX_Base('TCPIP::10.112.1.116', False, True)
print(f'CMX Base IND: {cmx_base.utilities.idn_string}')
print(f'CMX Instrument options:\n{",".join(cmx_base.utilities.instrument_options)}')
cmx_base.utilities.visa_timeout = 5000  # default is 10000

# Sends OPC after each command
cmx_base.utilities.opc_query_after_write = False
# Checks for syst:err? after each command / query - default value after init is True
cmx_base.utilities.instrument_status_checking = True

# Self-test
self_test = cmx_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')
# Reference Frequency Source
cmx_base.system.reference.frequency.source_set(base_enums.SourceIntExt.INTernal)

# CMX RsCMPX_Gprf Init - reuse the session of the cmx_base, rather than creating another␣
↪one
cmx_gprf = RsCMPX_Gprf.from_existing_session(cmx_base)
cmx_gprf.utilities.visa_timeout = 5000

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmx_gprf.reliability.ExceptionOnError = True  # default is 10000


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


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

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

# Close the sessions
cmx_gprf.close()
cmx_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 (for big files transfer)
- Multithreading session locking - you can use multiple threads talking to one instrument at the same time
- Logging feature tailored for SCPI communication - different for binary and ascii data

2.2 Installation

RsCMPX_LteMeas 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 RsCMPX_LteMeas`

Option 2 - Installing in Pycharm

- In Pycharm Menu `File->Settings->Project->Project Interpreter` click on the '+' button on the top left (the last PyCharm version)
- Type `RsCMPX_LteMeas` 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 6 step for installing the RsCMPX_LteMeas offline:

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCMPX_LteMeas needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCMPX_LteMeas package to your computer from the pypi.org: https://pypi.org/project/RsCMPX_LteMeas/#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\RsCMPX_LteMeas-5.0.70.14.tar`

2.3 Finding Available Instruments

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

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

from RsCMPX_LteMeas import *

# Use the instr_list string items as resource names in the RsCMPX_LteMeas constructor
instr_list = RsCMPX_LteMeas.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 RsCMPX_LteMeas 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 = RsCMPX_LteMeas.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

2.4 Initiating Instrument Session

RsCMPX_LteMeas 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 RsCMPX_LteMeas object. Below, is a simple Hello World example. Different resource names are examples for different physical interfaces.

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

- Installed RsCMPX_LteMeas Python module Version 5.0.70 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCMPX_LteMeas import *

# A good practice is to assure that you have a certain minimum version installed
RsCMPX_LteMeas.assert_minimum_version('5.0.70')
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 = RsCMPX_LteMeas(resource_string_1)

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f'RsCMPX_LteMeas 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 2023.

Do not care about specialty of each session kind; RsCMPX_LteMeas 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 = RsCMPX_LteMeas('TCPIP::192.168.56.101::hislip0', id_query=True, reset=True)
```

- Setting `id_query` to True (default is True) checks, whether your instrument can be used with the RsCMPX_LteMeas 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 RsCMPX_LteMeas allows you to choose which VISA to use:

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

from RsCMPX_LteMeas import *

# Force use of the Rs Visa. For NI Visa, use the "SelectVisa='ni'"
driver = RsCMPX_LteMeas('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, RsCMPX_LteMeas has something for you too - **no Visa installation raw LAN socket**:

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

(continues on next page)

(continued from previous page)

```
from RsCMPX_LteMeas import *

driver = RsCMPX_LteMeas('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 = RsCMPX_LteMeas('TCPIP::192.168.56.101::hislip0', True, True, "Simulate=True")
```

More `option_string` tokens are separated by comma:

```
driver = RsCMPX_LteMeas('TCPIP::192.168.56.101::hislip0', 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 RsCMPX_LteMeas objects:

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

from RsCMPX_LteMeas import *

driver1 = RsCMPX_LteMeas('TCPIP::192.168.56.101::INSTR', True, True)
driver2 = RsCMPX_LteMeas.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}')
```

(continues on next page)

(continued from previous page)

```
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.

2.5 Plain SCPI Communication

After you have opened the session, you can use the instrument-specific part described in the RsCMPX_LteMeas 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 RsCMPX_LteMeas import *

driver = RsCMPX_LteMeas('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 RsCMPX_LteMeas import *

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

# 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 RsCMPX_LteMeas raises an exception. Speaking of exceptions, an important feature of the RsCMPX_LteMeas 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 RsCMPX_LteMeas import *

driver = RsCMPX_LteMeas('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
```

2.6 Error Checking

RsCMPX_LteMeas 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.

2.7 Exception Handling

The base class for all the exceptions raised by the RsCMPX_LteMeas 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 RsCMPX_LteMeas import *

driver = None
# Try-catch for initialization. If an error occures, the ResourceError is raised
try:
    driver = RsCMPX_LteMeas('TCPIP::10.112.1.179::hislip0')
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 RsCMPX_LteMeas exceptions
    print(e.args[0])
    print('Some other RsCMPX_LteMeas 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.

2.8 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 RsCMPX_LteMeas, 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 RsCMPX_LteMeas 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')
```

2.9 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")
```

2.10 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 RsCMPX_LteMeas has this covered. And, this feature is quite universal - not just for big files transfer, but for any data in both directions.

RsCMPX_LteMeas 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 RsCMPX_LteMeas 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 = RsCMPX_LteMeas('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 RsCMPX_LteMeas 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
```

2.11 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, RsCMPX_LteMeas 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 RsCMPX_LteMeas object
"""

import threading
from RsCMPX_LteMeas import *
```

(continues on next page)

(continued from previous page)

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


driver = RsCMPX_LteMeas('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 RsCMPX_LteMeas objects with shared session
"""

import threading
from RsCMPX_LteMeas import *


def execute(session: RsCMPX_LteMeas, 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 = RsCMPX_LteMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_LteMeas.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 RsCMPX_LteMeas takes care of it for you. The text below describes this scenario.

Run the following example:

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

import threading
from RsCMPX_LteMeas import *


def execute(session: RsCMPX_LteMeas, 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 = RsCMPX_LteMeas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_LteMeas('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.

2.12 Logging

Yes, the logging again. This one is tailored for instrument communication. You will appreciate such handy feature when you troubleshoot your program, or just want to protocol the SCPI communication for your test reports.

What can you actually do with the logger?

- Write SCPI communication to a stream-like object, for example console or file, or both simultaneously
- Log only errors and skip problem-free parts; this way you avoid going through thousands lines of texts
- Investigate duration of certain operations to optimize your program's performance
- Log custom messages from your program

Let us take this basic example:

```
"""
Basic logging example to the console
"""

from RsCMPX_LteMeas import *

driver = RsCMPX_LteMeas('TCPIP::192.168.1.101::INSTR')
```

(continues on next page)

(continued from previous page)

```
# Switch ON logging to the console.
driver.utilities.logger.log_to_console = True
driver.utilities.logger.mode = LoggingMode.On
driver.utilities.reset()

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

Console output:

```
10:29:10.819     TCPIP::192.168.1.101::INSTR     0.976 ms  Write: *RST
10:29:10.819     TCPIP::192.168.1.101::INSTR  1884.985 ms  Status check: OK
10:29:12.704     TCPIP::192.168.1.101::INSTR     0.983 ms  Query OPC: 1
10:29:12.705     TCPIP::192.168.1.101::INSTR     2.892 ms  Clear status: OK
10:29:12.708     TCPIP::192.168.1.101::INSTR     3.905 ms  Status check: OK
10:29:12.712     TCPIP::192.168.1.101::INSTR     1.952 ms  Close: Closing session
```

The columns of the log are aligned for better reading. Columns meaning:

- (1) Start time of the operation
- (2) Device resource name (you can set an alias)
- (3) Duration of the operation
- (4) Log entry

Tip: You can customize the logging format with `set_format_string()`, and set the maximum log entry length with the properties:

- `abbreviated_max_len_ascii`
- `abbreviated_max_len_bin`
- `abbreviated_max_len_list`

See the full logger help *here*.

Notice the SCPI communication starts from the line `driver.utilities.reset()`. If you want to log the initialization of the session as well, you have to switch the logging ON already in the constructor:

```
driver = RsCMPX_LteMeas('TCPIP::192.168.56.101::hislip0', options='LoggingMode=On')
```

Parallel to the console logging, you can log to a general stream. Do not fear the programmer's jargon'... under the term **stream** you can just imagine a file. To be a little more technical, a stream in Python is any object that has two methods: `write()` and `flush()`. This example opens a file and sets it as logging target:

```
"""
Example of logging to a file
"""

from RsCMPX_LteMeas import *

driver = RsCMPX_LteMeas('TCPIP::192.168.1.101::INSTR')
```

(continues on next page)

(continued from previous page)

```
# We also want to log to the console.
driver.utilities.logger.log_to_console = True

# Logging target is our file
file = open(r'c:\temp\my_file.txt', 'w')
driver.utilities.logger.set_logging_target(file)
driver.utilities.logger.mode = LoggingMode.On

# Instead of the 'TCPIP::192.168.1.101::INSTR', show 'MyDevice'
driver.utilities.logger.device_name = 'MyDevice'

# Custom user entry
driver.utilities.logger.info_raw('----- This is my custom log entry. ---- ')

driver.utilities.reset()

# Close the session
driver.close()

# Close the log file
file.close()
```

Tip: To make the log more compact, you can skip all the lines with `Status check: OK`:

```
driver.utilities.logger.log_status_check_ok = False
```

Hint: You can share the logging file between multiple sessions. In such case, remember to close the file only after you have stopped logging in all your sessions, otherwise you get a log write error.

For logging to a UDP port in addition to other log targets, use one of the lines:

```
driver.utilities.logger.log_to_udp = True
driver.utilities.logger.log_to_console_and_udp = True
```

You can select the UDP port to log to, the default is 49200:

```
driver.utilities.logger.udp_port = 49200
```

Another cool feature is logging only errors. To make this mode usefull for troubleshooting, you also want to see the circumstances which lead to the errors. Each driver elementary operation, for example, `write_str()`, can generate a group of log entries - let us call them **Segment**. In the logging mode `Errors`, a whole segment is logged only if at least one entry of the segment is an error.

The script below demonstrates this feature. We use a direct SCPI communication to send a misspelled SCPI command *CLS, which leads to instrument status error:

```
"""
Logging example to the console with only errors logged
"""
```

(continues on next page)

(continued from previous page)

```
from RsCMPX_LteMeas import *

driver = RsCMPX_LteMeas('TCPIP::192.168.1.101::INSTR', options='LoggingMode=Errors')

# Switch ON logging to the console.
driver.utilities.logger.log_to_console = True

# Reset will not be logged, since no error occurred there
driver.utilities.reset()

# Now a misspelled command.
driver.utilities.write('*CLaS')

# A good command again, no logging here
idn = driver.utilities.query('*IDN?')

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

Console output:

```
12:11:02.879 TCPIP::192.168.1.101::INSTR     0.976 ms  Write string: *CLaS
12:11:02.879 TCPIP::192.168.1.101::INSTR     6.833 ms  Status check: StatusException:
                                                       Instrument error detected: Undefined header;
↪*CLaS
```

Notice the following:

- Although the operation **Write string: *CLaS** finished without an error, it is still logged, because it provides the context for the actual error which occurred during the status checking right after.
- No other log entries are present, including the session initialization and close, because they were all error-free.

CHAPTER

THREE

ENUMS

3.1 Band

```
# First value:
value = enums.Band.OB1
# Last value:
value = enums.Band.OB9
# All values (63x):
OB1 | OB10 | OB11 | OB12 | OB13 | OB14 | OB15 | OB16
OB17 | OB18 | OB19 | OB2 | OB20 | OB21 | OB22 | OB23
OB24 | OB25 | OB250 | OB26 | OB27 | OB28 | OB3 | OB30
OB31 | OB33 | OB34 | OB35 | OB36 | OB37 | OB38 | OB39
OB4 | OB40 | OB41 | OB42 | OB43 | OB44 | OB45 | OB46
OB47 | OB48 | OB49 | OB5 | OB50 | OB51 | OB52 | OB53
OB6 | OB65 | OB66 | OB68 | OB7 | OB70 | OB71 | OB72
OB73 | OB74 | OB8 | OB85 | OB87 | OB88 | OB9
```

3.2 BandwidthNarrow

```
# Example value:
value = enums.BandwidthNarrow.M010
# All values (4x):
M010 | M020 | M040 | M080
```

3.3 CarrAggrLocalOscLocation

```
# Example value:
value = enums.CarrAggrLocalOscLocation.AUTO
# All values (3x):
AUTO | CACB | CECC
```

3.4 CarrAggrMaping

```
# First value:
value = enums.CarrAggrMaping.INV
# Last value:
value = enums.CarrAggrMaping.SCC7
# All values (9x):
INV | PCC | SCC1 | SCC2 | SCC3 | SCC4 | SCC5 | SCC6
SCC7
```

3.5 CarrAggrMode

```
# Example value:
value = enums.CarrAggrMode.ICD
# All values (4x):
ICD | ICE | INTRaband | OFF
```

3.6 ChannelBandwidth

```
# Example value:
value = enums.ChannelBandwidth.B014
# All values (6x):
B014 | B030 | B050 | B100 | B150 | B200
```

3.7 ChannelTypeDetection

```
# Example value:
value = enums.ChannelTypeDetection.AUTO
# All values (3x):
AUTO | PUCCh | PUSCh
```

3.8 ChannelTypeVewFilter

```
# Example value:
value = enums.ChannelTypeVewFilter.OFF
# All values (4x):
OFF | ON | PUCCh | PUSCh
```

3.9 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
```

3.10 CyclicPrefix

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

3.11 FrameStructure

```
# Example value:
value = enums.FrameStructure.T1
# All values (2x):
T1 | T2
```

3.12 LaggingExclPeriod

```
# Example value:
value = enums.LaggingExclPeriod.MS05
# All values (3x):
MS05 | MS25 | OFF
```

3.13 LeadingExclPeriod

```
# Example value:
value = enums.LeadingExclPeriod.MS25
# All values (2x):
MS25 | OFF
```

3.14 ListMode

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

3.15 LocalOscLocation

```
# Example value:
value = enums.LocalOscLocation.CCB
# All values (2x):
CCB | CN
```

3.16 LowHigh

```
# Example value:
value = enums.LowHigh.HIGH
# All values (2x):
HIGH | LOW
```

3.17 MeasCarrier

```
# Example value:
value = enums.MeasCarrier.PCC
# All values (2x):
PCC | SCC1
```

3.18 MeasCarrierB

```
# Example value:
value = enums.MeasCarrierB.PCC
# All values (8x):
PCC | SCC1 | SCC2 | SCC3 | SCC4 | SCC5 | SCC6 | SCC7
```

3.19 MeasCarrierEnhanced

```
# Example value:
value = enums.MeasCarrierEnhanced.CC1
# All values (4x):
CC1 | CC2 | CC3 | CC4
```

3.20 MeasFilter

```
# Example value:
value = enums.MeasFilter.BANDpass
# All values (2x):
BANDpass | GAUSs
```

3.21 MeasurementMode

```
# Example value:
value = enums.MeasurementMode.MELMode
# All values (3x):
MELMode | NORMal | TMODe
```

3.22 MeasureSlot

```
# Example value:
value = enums.MeasureSlot.ALL
# All values (3x):
ALL | MS0 | MS1
```

3.23 MevAcquisitionMode

```
# Example value:
value = enums.MevAcquisitionMode.SLOT
# All values (2x):
SLOT | SUBFrame
```

3.24 Mode

```
# Example value:
value = enums.Mode.FDD
# All values (2x):
FDD | TDD
```

3.25 ModScheme

```
# Example value:
value = enums.ModScheme.AUTO
# All values (5x):
AUTO | Q16 | Q256 | Q64 | QPSK
```

3.26 Modulation

```
# Example value:
value = enums.Modulation.Q16
# All values (4x):
Q16 | Q256 | Q64 | QPSK
```

3.27 NetworkSharing

```
# Example value:
value = enums.NetworkSharing.FSHared
# All values (3x):
FSHared | NSHared | OCONnection
```

3.28 NetworkSigValue

```
# First value:
value = enums.NetworkSigValue.NS01
# Last value:
value = enums.NetworkSigValue.NS32
# All values (32x):
NS01 | NS02 | NS03 | NS04 | NS05 | NS06 | NS07 | NS08
NS09 | NS10 | NS11 | NS12 | NS13 | NS14 | NS15 | NS16
NS17 | NS18 | NS19 | NS20 | NS21 | NS22 | NS23 | NS24
NS25 | NS26 | NS27 | NS28 | NS29 | NS30 | NS31 | NS32
```

3.29 NetworkSigValueNoCarrAggr

```
# First value:
value = enums.NetworkSigValueNoCarrAggr.NS01
# Last value:
value = enums.NetworkSigValueNoCarrAggr.NS99
# All values (288x):
NS01 | NS02 | NS03 | NS04 | NS05 | NS06 | NS07 | NS08
NS09 | NS10 | NS100 | NS101 | NS102 | NS103 | NS104 | NS105
NS106 | NS107 | NS108 | NS109 | NS11 | NS110 | NS111 | NS112
NS113 | NS114 | NS115 | NS116 | NS117 | NS118 | NS119 | NS12
NS120 | NS121 | NS122 | NS123 | NS124 | NS125 | NS126 | NS127
NS128 | NS129 | NS13 | NS130 | NS131 | NS132 | NS133 | NS134
NS135 | NS136 | NS137 | NS138 | NS139 | NS14 | NS140 | NS141
NS142 | NS143 | NS144 | NS145 | NS146 | NS147 | NS148 | NS149
NS15 | NS150 | NS151 | NS152 | NS153 | NS154 | NS155 | NS156
NS157 | NS158 | NS159 | NS16 | NS160 | NS161 | NS162 | NS163
NS164 | NS165 | NS166 | NS167 | NS168 | NS169 | NS17 | NS170
NS171 | NS172 | NS173 | NS174 | NS175 | NS176 | NS177 | NS178
NS179 | NS18 | NS180 | NS181 | NS182 | NS183 | NS184 | NS185
NS186 | NS187 | NS188 | NS189 | NS19 | NS190 | NS191 | NS192
NS193 | NS194 | NS195 | NS196 | NS197 | NS198 | NS199 | NS20
NS200 | NS201 | NS202 | NS203 | NS204 | NS205 | NS206 | NS207
NS208 | NS209 | NS21 | NS210 | NS211 | NS212 | NS213 | NS214
NS215 | NS216 | NS217 | NS218 | NS219 | NS22 | NS220 | NS221
NS222 | NS223 | NS224 | NS225 | NS226 | NS227 | NS228 | NS229
NS23 | NS230 | NS231 | NS232 | NS233 | NS234 | NS235 | NS236
NS237 | NS238 | NS239 | NS24 | NS240 | NS241 | NS242 | NS243
NS244 | NS245 | NS246 | NS247 | NS248 | NS249 | NS25 | NS250
NS251 | NS252 | NS253 | NS254 | NS255 | NS256 | NS257 | NS258
NS259 | NS26 | NS260 | NS261 | NS262 | NS263 | NS264 | NS265
NS266 | NS267 | NS268 | NS269 | NS27 | NS270 | NS271 | NS272
NS273 | NS274 | NS275 | NS276 | NS277 | NS278 | NS279 | NS28
NS280 | NS281 | NS282 | NS283 | NS284 | NS285 | NS286 | NS287
NS288 | NS29 | NS30 | NS31 | NS32 | NS33 | NS34 | NS35
NS36 | NS37 | NS38 | NS39 | NS40 | NS41 | NS42 | NS43
NS44 | NS45 | NS46 | NS47 | NS48 | NS49 | NS50 | NS51
NS52 | NS53 | NS54 | NS55 | NS56 | NS57 | NS58 | NS59
NS60 | NS61 | NS62 | NS63 | NS64 | NS65 | NS66 | NS67
NS68 | NS69 | NS70 | NS71 | NS72 | NS73 | NS74 | NS75
NS76 | NS77 | NS78 | NS79 | NS80 | NS81 | NS82 | NS83
NS84 | NS85 | NS86 | NS87 | NS88 | NS89 | NS90 | NS91
NS92 | NS93 | NS94 | NS95 | NS96 | NS97 | NS98 | NS99
```

3.30 ParameterSetMode

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

3.31 Path

```
# Example value:
value = enums.Path.NETWork
# All values (2x):
NETWork | STANdalone
```

3.32 PeriodPreamble

```
# Example value:
value = enums.PeriodPreamble.MS05
# All values (3x):
MS05 | MS10 | MS20
```

3.33 PucchFormat

```
# Example value:
value = enums.PucchFormat.F1
# All values (7x):
F1 | F1A | F1B | F2 | F2A | F2B | F3
```

3.34 RbTableChannelType

```
# Example value:
value = enums.RbTableChannelType.DL
# All values (8x):
DL | NONE | PSBCh | PSCCh | PSSCh | PUCCh | PUSCh | SSUB
```

3.35 Rbw

```
# Example value:
value = enums.Rbw.K030
# All values (3x):
K030 | K100 | M1
```

3.36 RbwExtended

```
# Example value:
value = enums.RbwExtended.K030
# All values (6x):
K030 | K050 | K100 | K150 | K200 | M1
```

3.37 Repeat

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

3.38 ResourceState

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

3.39 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
```

3.40 RetriggerFlag

```
# Example value:
value = enums.RetriggerFlag.IFPNarrow
# All values (4x):
IFPNarrow | IFPower | OFF | ON
```

3.41 SegmentChannelTypeExtended

```
# Example value:
value = enums.SegmentChannelTypeExtended.AUTO
# All values (6x):
AUTO | PSBCh | PSCCh | PSSCh | PUCCh | PUSCh
```

3.42 SidelinkChannelType

```
# Example value:
value = enums.SidelinkChannelType.PSBCh
# All values (3x):
PSBCh | PSCCh | PSSCh
```

3.43 SignalSlope

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

3.44 SignalType

```
# Example value:
value = enums.SignalType.SL
# All values (2x):
SL | UL
```

3.45 StopCondition

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

3.46 SyncMode

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

3.47 TargetStateA

```
# Example value:
value = enums.TargetStateA.OFF
# All values (3x):
OFF | RDY | RUN
```

3.48 TargetSyncState

```
# Example value:
value = enums.TargetSyncState.ADJusted
# All values (2x):
ADJusted | PENDing
```

3.49 TimeMask

```
# Example value:
value = enums.TimeMask.GOO
# All values (3x):
GOO | PPSRs | SBLanking
```

3.50 TraceSelect

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

3.51 UplinkChannelType

```
# Example value:
value = enums.UplinkChannelType.PUCCh
# All values (2x):
PUCCh | PUSCh
```

3.52 ViewMev

```
# First value:
value = enums.ViewMev.ACLR
# Last value:
value = enums.ViewMev.TXM
# All values (15x):
ACLR | BLER | ESFLatness | EVMagnitude | EVMC | IEMissions | IQ | MERRor
OVERview | PDYNamics | PERRor | PMONitor | RBATable | SEMask | TXM
```

3.53 ViewPrach

```
# First value:
value = enums.ViewPrach.EVMagnitude
# Last value:
value = enums.ViewPrach.TXM
# All values (10x):
EVMagnitude | EVPReamble | EVSYmbol | IQ | MERRor | OVERview | PDYNamics | PERRor
PVPReamble | TXM
```

3.54 ViewSrs

```
# Example value:
value = enums.ViewSrs.PDYNamics
# All values (1x):
PDYNamics
```

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 AbsMarker

```
# First value:
value = repcap.AbsMarker.Nr1
# Values (2x):
Nr1 | Nr2
```

4.3 Area

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

4.4 CarrierComponent

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

4.5 CarrierComponentB

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

4.6 ChannelBw

```
# First value:
value = repcap.ChannelBw.Bw14
# Range:
Bw14 .. Bw200
# All values (6x):
Bw14 | Bw30 | Bw50 | Bw100 | Bw150 | Bw200
```

4.7 DeltaMarker

```
# First value:
value = repcap.DeltaMarker.Nr1
# Values (2x):
Nr1 | Nr2
```

4.8 Difference

```
# First value:
value = repcap.Difference.Nr1
# Values (2x):
Nr1 | Nr2
```

4.9 EutraBand

```
# First value:
value = repcap.EutraBand.Nr30
# Values (2x):
Nr30 | Nr50
```

4.10 FirstChannelBw

```
# First value:
value = repcap.FirstChannelBw.Bw100
# Values (3x):
Bw100 | Bw150 | Bw200
```

4.11 Limit

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

4.12 MaxRange

```
# First value:
value = repcap.MaxRange.Nr1
# Values (2x):
Nr1 | Nr2
```

4.13 MinRange

```
# First value:
value = repcap.MinRange.Nr1
# Values (2x):
Nr1 | Nr2
```

4.14 Preamble

```
# First value:
value = repcap.Preamble.Nr1
# Range:
Nr1 .. Nr400
# All values (400x):
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
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
```

(continues on next page)

(continued from previous page)

```
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
```

4.15 PreambleFormat

```
# First value:
value = repcap.PreambleFormat.Fmt1
# Range:
Fmt1 .. Fmt5
# All values (5x):
Fmt1 | Fmt2 | Fmt3 | Fmt4 | Fmt5
```

4.16 QAMmodOrder

```
# First value:
value = repcap.QAMmodOrder.Qam16
# Values (3x):
Qam16 | Qam64 | Qam256
```

4.17 RBcount

```
# First value:
value = repcap.RBcount.Nr1
# Values (2x):
Nr1 | Nr2
```

4.18 RBoffset

```
# First value:
value = repcap.RBoffset.Nr1
# Values (2x):
Nr1 | Nr2
```

4.19 RBWkHz

```
# First value:
value = repcap.RBWkHz.Rbw30
# Range:
Rbw30 .. Rbw1000
# All values (6x):
Rbw30 | Rbw50 | Rbw100 | Rbw150 | Rbw200 | Rbw1000
```

4.20 Ripple

```
# First value:
value = repcap.Ripple.Nr1
# Values (2x):
Nr1 | Nr2
```

4.21 SecondaryCC

```
# First value:
value = repcap.SecondaryCC.CC1
# Range:
CC1 .. CC7
# All values (7x):
CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7
```

4.22 SecondChannelBw

```
# First value:
value = repcap.SecondChannelBw.Bw50
# Values (4x):
Bw50 | Bw100 | Bw150 | Bw200
```

4.23 Segment

```
# First value:
value = repcap.Segment.Nr1
# Range:
Nr1 .. Nr2000
# All values (2000x):
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
```

(continues on next page)

(continued from previous page)

```
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
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
```

(continues on next page)

(continued from previous page)

```
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
Nr513 | Nr514 | Nr515 | Nr516 | Nr517 | Nr518 | Nr519 | Nr520
Nr521 | Nr522 | Nr523 | Nr524 | Nr525 | Nr526 | Nr527 | Nr528
Nr529 | Nr530 | Nr531 | Nr532 | Nr533 | Nr534 | Nr535 | Nr536
Nr537 | Nr538 | Nr539 | Nr540 | Nr541 | Nr542 | Nr543 | Nr544
Nr545 | Nr546 | Nr547 | Nr548 | Nr549 | Nr550 | Nr551 | Nr552
Nr553 | Nr554 | Nr555 | Nr556 | Nr557 | Nr558 | Nr559 | Nr560
Nr561 | Nr562 | Nr563 | Nr564 | Nr565 | Nr566 | Nr567 | Nr568
Nr569 | Nr570 | Nr571 | Nr572 | Nr573 | Nr574 | Nr575 | Nr576
Nr577 | Nr578 | Nr579 | Nr580 | Nr581 | Nr582 | Nr583 | Nr584
Nr585 | Nr586 | Nr587 | Nr588 | Nr589 | Nr590 | Nr591 | Nr592
Nr593 | Nr594 | Nr595 | Nr596 | Nr597 | Nr598 | Nr599 | Nr600
Nr601 | Nr602 | Nr603 | Nr604 | Nr605 | Nr606 | Nr607 | Nr608
Nr609 | Nr610 | Nr611 | Nr612 | Nr613 | Nr614 | Nr615 | Nr616
Nr617 | Nr618 | Nr619 | Nr620 | Nr621 | Nr622 | Nr623 | Nr624
Nr625 | Nr626 | Nr627 | Nr628 | Nr629 | Nr630 | Nr631 | Nr632
Nr633 | Nr634 | Nr635 | Nr636 | Nr637 | Nr638 | Nr639 | Nr640
Nr641 | Nr642 | Nr643 | Nr644 | Nr645 | Nr646 | Nr647 | Nr648
Nr649 | Nr650 | Nr651 | Nr652 | Nr653 | Nr654 | Nr655 | Nr656
Nr657 | Nr658 | Nr659 | Nr660 | Nr661 | Nr662 | Nr663 | Nr664
Nr665 | Nr666 | Nr667 | Nr668 | Nr669 | Nr670 | Nr671 | Nr672
Nr673 | Nr674 | Nr675 | Nr676 | Nr677 | Nr678 | Nr679 | Nr680
Nr681 | Nr682 | Nr683 | Nr684 | Nr685 | Nr686 | Nr687 | Nr688
Nr689 | Nr690 | Nr691 | Nr692 | Nr693 | Nr694 | Nr695 | Nr696
Nr697 | Nr698 | Nr699 | Nr700 | Nr701 | Nr702 | Nr703 | Nr704
Nr705 | Nr706 | Nr707 | Nr708 | Nr709 | Nr710 | Nr711 | Nr712
Nr713 | Nr714 | Nr715 | Nr716 | Nr717 | Nr718 | Nr719 | Nr720
Nr721 | Nr722 | Nr723 | Nr724 | Nr725 | Nr726 | Nr727 | Nr728
Nr729 | Nr730 | Nr731 | Nr732 | Nr733 | Nr734 | Nr735 | Nr736
Nr737 | Nr738 | Nr739 | Nr740 | Nr741 | Nr742 | Nr743 | Nr744
Nr745 | Nr746 | Nr747 | Nr748 | Nr749 | Nr750 | Nr751 | Nr752
Nr753 | Nr754 | Nr755 | Nr756 | Nr757 | Nr758 | Nr759 | Nr760
Nr761 | Nr762 | Nr763 | Nr764 | Nr765 | Nr766 | Nr767 | Nr768
Nr769 | Nr770 | Nr771 | Nr772 | Nr773 | Nr774 | Nr775 | Nr776
Nr777 | Nr778 | Nr779 | Nr780 | Nr781 | Nr782 | Nr783 | Nr784
Nr785 | Nr786 | Nr787 | Nr788 | Nr789 | Nr790 | Nr791 | Nr792
Nr793 | Nr794 | Nr795 | Nr796 | Nr797 | Nr798 | Nr799 | Nr800
Nr801 | Nr802 | Nr803 | Nr804 | Nr805 | Nr806 | Nr807 | Nr808
Nr809 | Nr810 | Nr811 | Nr812 | Nr813 | Nr814 | Nr815 | Nr816
Nr817 | Nr818 | Nr819 | Nr820 | Nr821 | Nr822 | Nr823 | Nr824
Nr825 | Nr826 | Nr827 | Nr828 | Nr829 | Nr830 | Nr831 | Nr832
Nr833 | Nr834 | Nr835 | Nr836 | Nr837 | Nr838 | Nr839 | Nr840
Nr841 | Nr842 | Nr843 | Nr844 | Nr845 | Nr846 | Nr847 | Nr848
Nr849 | Nr850 | Nr851 | Nr852 | Nr853 | Nr854 | Nr855 | Nr856
Nr857 | Nr858 | Nr859 | Nr860 | Nr861 | Nr862 | Nr863 | Nr864
```

(continues on next page)

(continued from previous page)

```
Nr865 | Nr866 | Nr867 | Nr868 | Nr869 | Nr870 | Nr871 | Nr872
Nr873 | Nr874 | Nr875 | Nr876 | Nr877 | Nr878 | Nr879 | Nr880
Nr881 | Nr882 | Nr883 | Nr884 | Nr885 | Nr886 | Nr887 | Nr888
Nr889 | Nr890 | Nr891 | Nr892 | Nr893 | Nr894 | Nr895 | Nr896
Nr897 | Nr898 | Nr899 | Nr900 | Nr901 | Nr902 | Nr903 | Nr904
Nr905 | Nr906 | Nr907 | Nr908 | Nr909 | Nr910 | Nr911 | Nr912
Nr913 | Nr914 | Nr915 | Nr916 | Nr917 | Nr918 | Nr919 | Nr920
Nr921 | Nr922 | Nr923 | Nr924 | Nr925 | Nr926 | Nr927 | Nr928
Nr929 | Nr930 | Nr931 | Nr932 | Nr933 | Nr934 | Nr935 | Nr936
Nr937 | Nr938 | Nr939 | Nr940 | Nr941 | Nr942 | Nr943 | Nr944
Nr945 | Nr946 | Nr947 | Nr948 | Nr949 | Nr950 | Nr951 | Nr952
Nr953 | Nr954 | Nr955 | Nr956 | Nr957 | Nr958 | Nr959 | Nr960
Nr961 | Nr962 | Nr963 | Nr964 | Nr965 | Nr966 | Nr967 | Nr968
Nr969 | Nr970 | Nr971 | Nr972 | Nr973 | Nr974 | Nr975 | Nr976
Nr977 | Nr978 | Nr979 | Nr980 | Nr981 | Nr982 | Nr983 | Nr984
Nr985 | Nr986 | Nr987 | Nr988 | Nr989 | Nr990 | Nr991 | Nr992
Nr993 | Nr994 | Nr995 | Nr996 | Nr997 | Nr998 | Nr999 | Nr1000
Nr1001 | Nr1002 | Nr1003 | Nr1004 | Nr1005 | Nr1006 | Nr1007 | Nr1008
Nr1009 | Nr1010 | Nr1011 | Nr1012 | Nr1013 | Nr1014 | Nr1015 | Nr1016
Nr1017 | Nr1018 | Nr1019 | Nr1020 | Nr1021 | Nr1022 | Nr1023 | Nr1024
Nr1025 | Nr1026 | Nr1027 | Nr1028 | Nr1029 | Nr1030 | Nr1031 | Nr1032
Nr1033 | Nr1034 | Nr1035 | Nr1036 | Nr1037 | Nr1038 | Nr1039 | Nr1040
Nr1041 | Nr1042 | Nr1043 | Nr1044 | Nr1045 | Nr1046 | Nr1047 | Nr1048
Nr1049 | Nr1050 | Nr1051 | Nr1052 | Nr1053 | Nr1054 | Nr1055 | Nr1056
Nr1057 | Nr1058 | Nr1059 | Nr1060 | Nr1061 | Nr1062 | Nr1063 | Nr1064
Nr1065 | Nr1066 | Nr1067 | Nr1068 | Nr1069 | Nr1070 | Nr1071 | Nr1072
Nr1073 | Nr1074 | Nr1075 | Nr1076 | Nr1077 | Nr1078 | Nr1079 | Nr1080
Nr1081 | Nr1082 | Nr1083 | Nr1084 | Nr1085 | Nr1086 | Nr1087 | Nr1088
Nr1089 | Nr1090 | Nr1091 | Nr1092 | Nr1093 | Nr1094 | Nr1095 | Nr1096
Nr1097 | Nr1098 | Nr1099 | Nr1100 | Nr1101 | Nr1102 | Nr1103 | Nr1104
Nr1105 | Nr1106 | Nr1107 | Nr1108 | Nr1109 | Nr1110 | Nr1111 | Nr1112
Nr1113 | Nr1114 | Nr1115 | Nr1116 | Nr1117 | Nr1118 | Nr1119 | Nr1120
Nr1121 | Nr1122 | Nr1123 | Nr1124 | Nr1125 | Nr1126 | Nr1127 | Nr1128
Nr1129 | Nr1130 | Nr1131 | Nr1132 | Nr1133 | Nr1134 | Nr1135 | Nr1136
Nr1137 | Nr1138 | Nr1139 | Nr1140 | Nr1141 | Nr1142 | Nr1143 | Nr1144
Nr1145 | Nr1146 | Nr1147 | Nr1148 | Nr1149 | Nr1150 | Nr1151 | Nr1152
Nr1153 | Nr1154 | Nr1155 | Nr1156 | Nr1157 | Nr1158 | Nr1159 | Nr1160
Nr1161 | Nr1162 | Nr1163 | Nr1164 | Nr1165 | Nr1166 | Nr1167 | Nr1168
Nr1169 | Nr1170 | Nr1171 | Nr1172 | Nr1173 | Nr1174 | Nr1175 | Nr1176
Nr1177 | Nr1178 | Nr1179 | Nr1180 | Nr1181 | Nr1182 | Nr1183 | Nr1184
Nr1185 | Nr1186 | Nr1187 | Nr1188 | Nr1189 | Nr1190 | Nr1191 | Nr1192
Nr1193 | Nr1194 | Nr1195 | Nr1196 | Nr1197 | Nr1198 | Nr1199 | Nr1200
Nr1201 | Nr1202 | Nr1203 | Nr1204 | Nr1205 | Nr1206 | Nr1207 | Nr1208
Nr1209 | Nr1210 | Nr1211 | Nr1212 | Nr1213 | Nr1214 | Nr1215 | Nr1216
Nr1217 | Nr1218 | Nr1219 | Nr1220 | Nr1221 | Nr1222 | Nr1223 | Nr1224
Nr1225 | Nr1226 | Nr1227 | Nr1228 | Nr1229 | Nr1230 | Nr1231 | Nr1232
Nr1233 | Nr1234 | Nr1235 | Nr1236 | Nr1237 | Nr1238 | Nr1239 | Nr1240
Nr1241 | Nr1242 | Nr1243 | Nr1244 | Nr1245 | Nr1246 | Nr1247 | Nr1248
Nr1249 | Nr1250 | Nr1251 | Nr1252 | Nr1253 | Nr1254 | Nr1255 | Nr1256
Nr1257 | Nr1258 | Nr1259 | Nr1260 | Nr1261 | Nr1262 | Nr1263 | Nr1264
Nr1265 | Nr1266 | Nr1267 | Nr1268 | Nr1269 | Nr1270 | Nr1271 | Nr1272
Nr1273 | Nr1274 | Nr1275 | Nr1276 | Nr1277 | Nr1278 | Nr1279 | Nr1280
```

(continues on next page)

(continued from previous page)

```
Nr1281 | Nr1282 | Nr1283 | Nr1284 | Nr1285 | Nr1286 | Nr1287 | Nr1288
Nr1289 | Nr1290 | Nr1291 | Nr1292 | Nr1293 | Nr1294 | Nr1295 | Nr1296
Nr1297 | Nr1298 | Nr1299 | Nr1300 | Nr1301 | Nr1302 | Nr1303 | Nr1304
Nr1305 | Nr1306 | Nr1307 | Nr1308 | Nr1309 | Nr1310 | Nr1311 | Nr1312
Nr1313 | Nr1314 | Nr1315 | Nr1316 | Nr1317 | Nr1318 | Nr1319 | Nr1320
Nr1321 | Nr1322 | Nr1323 | Nr1324 | Nr1325 | Nr1326 | Nr1327 | Nr1328
Nr1329 | Nr1330 | Nr1331 | Nr1332 | Nr1333 | Nr1334 | Nr1335 | Nr1336
Nr1337 | Nr1338 | Nr1339 | Nr1340 | Nr1341 | Nr1342 | Nr1343 | Nr1344
Nr1345 | Nr1346 | Nr1347 | Nr1348 | Nr1349 | Nr1350 | Nr1351 | Nr1352
Nr1353 | Nr1354 | Nr1355 | Nr1356 | Nr1357 | Nr1358 | Nr1359 | Nr1360
Nr1361 | Nr1362 | Nr1363 | Nr1364 | Nr1365 | Nr1366 | Nr1367 | Nr1368
Nr1369 | Nr1370 | Nr1371 | Nr1372 | Nr1373 | Nr1374 | Nr1375 | Nr1376
Nr1377 | Nr1378 | Nr1379 | Nr1380 | Nr1381 | Nr1382 | Nr1383 | Nr1384
Nr1385 | Nr1386 | Nr1387 | Nr1388 | Nr1389 | Nr1390 | Nr1391 | Nr1392
Nr1393 | Nr1394 | Nr1395 | Nr1396 | Nr1397 | Nr1398 | Nr1399 | Nr1400
Nr1401 | Nr1402 | Nr1403 | Nr1404 | Nr1405 | Nr1406 | Nr1407 | Nr1408
Nr1409 | Nr1410 | Nr1411 | Nr1412 | Nr1413 | Nr1414 | Nr1415 | Nr1416
Nr1417 | Nr1418 | Nr1419 | Nr1420 | Nr1421 | Nr1422 | Nr1423 | Nr1424
Nr1425 | Nr1426 | Nr1427 | Nr1428 | Nr1429 | Nr1430 | Nr1431 | Nr1432
Nr1433 | Nr1434 | Nr1435 | Nr1436 | Nr1437 | Nr1438 | Nr1439 | Nr1440
Nr1441 | Nr1442 | Nr1443 | Nr1444 | Nr1445 | Nr1446 | Nr1447 | Nr1448
Nr1449 | Nr1450 | Nr1451 | Nr1452 | Nr1453 | Nr1454 | Nr1455 | Nr1456
Nr1457 | Nr1458 | Nr1459 | Nr1460 | Nr1461 | Nr1462 | Nr1463 | Nr1464
Nr1465 | Nr1466 | Nr1467 | Nr1468 | Nr1469 | Nr1470 | Nr1471 | Nr1472
Nr1473 | Nr1474 | Nr1475 | Nr1476 | Nr1477 | Nr1478 | Nr1479 | Nr1480
Nr1481 | Nr1482 | Nr1483 | Nr1484 | Nr1485 | Nr1486 | Nr1487 | Nr1488
Nr1489 | Nr1490 | Nr1491 | Nr1492 | Nr1493 | Nr1494 | Nr1495 | Nr1496
Nr1497 | Nr1498 | Nr1499 | Nr1500 | Nr1501 | Nr1502 | Nr1503 | Nr1504
Nr1505 | Nr1506 | Nr1507 | Nr1508 | Nr1509 | Nr1510 | Nr1511 | Nr1512
Nr1513 | Nr1514 | Nr1515 | Nr1516 | Nr1517 | Nr1518 | Nr1519 | Nr1520
Nr1521 | Nr1522 | Nr1523 | Nr1524 | Nr1525 | Nr1526 | Nr1527 | Nr1528
Nr1529 | Nr1530 | Nr1531 | Nr1532 | Nr1533 | Nr1534 | Nr1535 | Nr1536
Nr1537 | Nr1538 | Nr1539 | Nr1540 | Nr1541 | Nr1542 | Nr1543 | Nr1544
Nr1545 | Nr1546 | Nr1547 | Nr1548 | Nr1549 | Nr1550 | Nr1551 | Nr1552
Nr1553 | Nr1554 | Nr1555 | Nr1556 | Nr1557 | Nr1558 | Nr1559 | Nr1560
Nr1561 | Nr1562 | Nr1563 | Nr1564 | Nr1565 | Nr1566 | Nr1567 | Nr1568
Nr1569 | Nr1570 | Nr1571 | Nr1572 | Nr1573 | Nr1574 | Nr1575 | Nr1576
Nr1577 | Nr1578 | Nr1579 | Nr1580 | Nr1581 | Nr1582 | Nr1583 | Nr1584
Nr1585 | Nr1586 | Nr1587 | Nr1588 | Nr1589 | Nr1590 | Nr1591 | Nr1592
Nr1593 | Nr1594 | Nr1595 | Nr1596 | Nr1597 | Nr1598 | Nr1599 | Nr1600
Nr1601 | Nr1602 | Nr1603 | Nr1604 | Nr1605 | Nr1606 | Nr1607 | Nr1608
Nr1609 | Nr1610 | Nr1611 | Nr1612 | Nr1613 | Nr1614 | Nr1615 | Nr1616
Nr1617 | Nr1618 | Nr1619 | Nr1620 | Nr1621 | Nr1622 | Nr1623 | Nr1624
Nr1625 | Nr1626 | Nr1627 | Nr1628 | Nr1629 | Nr1630 | Nr1631 | Nr1632
Nr1633 | Nr1634 | Nr1635 | Nr1636 | Nr1637 | Nr1638 | Nr1639 | Nr1640
Nr1641 | Nr1642 | Nr1643 | Nr1644 | Nr1645 | Nr1646 | Nr1647 | Nr1648
Nr1649 | Nr1650 | Nr1651 | Nr1652 | Nr1653 | Nr1654 | Nr1655 | Nr1656
Nr1657 | Nr1658 | Nr1659 | Nr1660 | Nr1661 | Nr1662 | Nr1663 | Nr1664
Nr1665 | Nr1666 | Nr1667 | Nr1668 | Nr1669 | Nr1670 | Nr1671 | Nr1672
Nr1673 | Nr1674 | Nr1675 | Nr1676 | Nr1677 | Nr1678 | Nr1679 | Nr1680
Nr1681 | Nr1682 | Nr1683 | Nr1684 | Nr1685 | Nr1686 | Nr1687 | Nr1688
Nr1689 | Nr1690 | Nr1691 | Nr1692 | Nr1693 | Nr1694 | Nr1695 | Nr1696
```

(continues on next page)

(continued from previous page)

```
Nr1697 | Nr1698 | Nr1699 | Nr1700 | Nr1701 | Nr1702 | Nr1703 | Nr1704
Nr1705 | Nr1706 | Nr1707 | Nr1708 | Nr1709 | Nr1710 | Nr1711 | Nr1712
Nr1713 | Nr1714 | Nr1715 | Nr1716 | Nr1717 | Nr1718 | Nr1719 | Nr1720
Nr1721 | Nr1722 | Nr1723 | Nr1724 | Nr1725 | Nr1726 | Nr1727 | Nr1728
Nr1729 | Nr1730 | Nr1731 | Nr1732 | Nr1733 | Nr1734 | Nr1735 | Nr1736
Nr1737 | Nr1738 | Nr1739 | Nr1740 | Nr1741 | Nr1742 | Nr1743 | Nr1744
Nr1745 | Nr1746 | Nr1747 | Nr1748 | Nr1749 | Nr1750 | Nr1751 | Nr1752
Nr1753 | Nr1754 | Nr1755 | Nr1756 | Nr1757 | Nr1758 | Nr1759 | Nr1760
Nr1761 | Nr1762 | Nr1763 | Nr1764 | Nr1765 | Nr1766 | Nr1767 | Nr1768
Nr1769 | Nr1770 | Nr1771 | Nr1772 | Nr1773 | Nr1774 | Nr1775 | Nr1776
Nr1777 | Nr1778 | Nr1779 | Nr1780 | Nr1781 | Nr1782 | Nr1783 | Nr1784
Nr1785 | Nr1786 | Nr1787 | Nr1788 | Nr1789 | Nr1790 | Nr1791 | Nr1792
Nr1793 | Nr1794 | Nr1795 | Nr1796 | Nr1797 | Nr1798 | Nr1799 | Nr1800
Nr1801 | Nr1802 | Nr1803 | Nr1804 | Nr1805 | Nr1806 | Nr1807 | Nr1808
Nr1809 | Nr1810 | Nr1811 | Nr1812 | Nr1813 | Nr1814 | Nr1815 | Nr1816
Nr1817 | Nr1818 | Nr1819 | Nr1820 | Nr1821 | Nr1822 | Nr1823 | Nr1824
Nr1825 | Nr1826 | Nr1827 | Nr1828 | Nr1829 | Nr1830 | Nr1831 | Nr1832
Nr1833 | Nr1834 | Nr1835 | Nr1836 | Nr1837 | Nr1838 | Nr1839 | Nr1840
Nr1841 | Nr1842 | Nr1843 | Nr1844 | Nr1845 | Nr1846 | Nr1847 | Nr1848
Nr1849 | Nr1850 | Nr1851 | Nr1852 | Nr1853 | Nr1854 | Nr1855 | Nr1856
Nr1857 | Nr1858 | Nr1859 | Nr1860 | Nr1861 | Nr1862 | Nr1863 | Nr1864
Nr1865 | Nr1866 | Nr1867 | Nr1868 | Nr1869 | Nr1870 | Nr1871 | Nr1872
Nr1873 | Nr1874 | Nr1875 | Nr1876 | Nr1877 | Nr1878 | Nr1879 | Nr1880
Nr1881 | Nr1882 | Nr1883 | Nr1884 | Nr1885 | Nr1886 | Nr1887 | Nr1888
Nr1889 | Nr1890 | Nr1891 | Nr1892 | Nr1893 | Nr1894 | Nr1895 | Nr1896
Nr1897 | Nr1898 | Nr1899 | Nr1900 | Nr1901 | Nr1902 | Nr1903 | Nr1904
Nr1905 | Nr1906 | Nr1907 | Nr1908 | Nr1909 | Nr1910 | Nr1911 | Nr1912
Nr1913 | Nr1914 | Nr1915 | Nr1916 | Nr1917 | Nr1918 | Nr1919 | Nr1920
Nr1921 | Nr1922 | Nr1923 | Nr1924 | Nr1925 | Nr1926 | Nr1927 | Nr1928
Nr1929 | Nr1930 | Nr1931 | Nr1932 | Nr1933 | Nr1934 | Nr1935 | Nr1936
Nr1937 | Nr1938 | Nr1939 | Nr1940 | Nr1941 | Nr1942 | Nr1943 | Nr1944
Nr1945 | Nr1946 | Nr1947 | Nr1948 | Nr1949 | Nr1950 | Nr1951 | Nr1952
Nr1953 | Nr1954 | Nr1955 | Nr1956 | Nr1957 | Nr1958 | Nr1959 | Nr1960
Nr1961 | Nr1962 | Nr1963 | Nr1964 | Nr1965 | Nr1966 | Nr1967 | Nr1968
Nr1969 | Nr1970 | Nr1971 | Nr1972 | Nr1973 | Nr1974 | Nr1975 | Nr1976
Nr1977 | Nr1978 | Nr1979 | Nr1980 | Nr1981 | Nr1982 | Nr1983 | Nr1984
Nr1985 | Nr1986 | Nr1987 | Nr1988 | Nr1989 | Nr1990 | Nr1991 | Nr1992
Nr1993 | Nr1994 | Nr1995 | Nr1996 | Nr1997 | Nr1998 | Nr1999 | Nr2000
```

4.24 Table

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

4.25 ThirdChannelBw

```
# First value:
value = repcap.ThirdChannelBw.Bw100
# Values (3x):
Bw100 | Bw150 | Bw200
```

4.26 UtraAdjChannel

```
# First value:
value = repcap.UtraAdjChannel.Ch1
# Values (2x):
Ch1 | Ch2
```

CHAPTER

FIVE

EXAMPLES

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

```
"""
# GitHub examples repository path: CMXP/Python/RsCmxp_xxx_ScpiPackages

Example on how to use the python RsCmx auto-generated instrument drivers for
RsCmpx_Base and RsCmpx_Gprf (Base and GPRF) in one script with shared VISA session.
"""

from RsCMPX_Base.RsCMPX_Base import RsCMPX_Base  # install from pypi.org
from RsCMPX_Base import enums as base_enums
from RsCMPX_Base import repcap as base_repcap

from RsCMPX_Gprf.RsCMPX_Gprf import RsCMPX_Gprf  # install from pypi.org
from RsCMPX_Gprf.CustomFiles.reliability import ReliabilityEventArgs
from RsCMPX_Gprf import enums as gprf_enums
from RsCMPX_Gprf import repcap as gprf_repcaps


# CMX Base init
cmx_base = RsCMPX_Base('TCPIP::10.112.1.116', False, True)
print(f'CMX Base IND: {cmx_base.utilities.idn_string}')
print(f'CMX Instrument options:\n{",".join(cmx_base.utilities.instrument_options)}')
cmx_base.utilities.visa_timeout = 5000  # default is 10000

# Sends OPC after each command
cmx_base.utilities.opc_query_after_write = False
# Checks for syst:err? after each command / query - default value after init is True
cmx_base.utilities.instrument_status_checking = True

# Self-test
self_test = cmx_base.utilities.self_test()
print(f'CMW self-test result: {self_test} - {"Passed" if self_test[0] == 0 else "Failed"}
↪"')
# Reference Frequency Source
cmx_base.system.reference.frequency.source_set(base_enums.SourceIntExt.INTernal)

# CMX RsCMPX_Gprf Init - reuse the session of the cmx_base, rather than creating another␣
↪one
cmx_gprf = RsCMPX_Gprf.from_existing_session(cmx_base)
```

(continues on next page)

(continued from previous page)

```
cmx_gprf.utilities.visa_timeout = 5000

# Driver's Interface reliability offers a convenient way of reacting on the return value␣
↪Reliability Indicator
cmx_gprf.reliability.ExceptionOnError = True  # default is 10000


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


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

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

# Close the sessions
cmx_gprf.close()
cmx_base.close()
```

CHAPTER

SIX

RSCMPX_LTEMEAS API STRUCTURE

Global RepCaps

```
driver = RsCMPX_LteMeas('TCPIP::192.168.2.101::hislip0')
# Instance range: Inst1 .. Inst16
rc = driver.repcap_instance_get()
driver.repcap_instance_set(repcap.Instance.Inst1)
```

class RsCMPX_LteMeas(*resource_name: str*, *id_query: bool = True*, *reset: bool = False*, *options: str = None*, *direct_session: object = None*)

1037 total commands, 4 Subgroups, 0 group commands

Initializes new RsCMPX_LteMeas 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=ivi` - forces usage of National Instruments Visa
- `QueryInstrumentStatus = False` - same as `driver.utilities.instrument_status_checking = False`. Default: `True`
- `WriteDelay = 20, ReadDelay = 5` - Introduces delay of 20ms before each write and 5ms before each read. Default: `0ms` for both
- `OpcWaitMode = OpcQuery` - mode for all the opc-synchronised write/reads. Other modes: StbPolling, StbPollingSlow, StbPollingSuperSlow. Default: `StbPolling`
- `AddTermCharToWriteBinBLock = True` - Adds one additional LF to the end of the binary data (some instruments require that). Default: `False`
- `AssureWriteWithTermChar = True` - Makes sure each command/query is terminated with termination character. Default: Interface dependent
- `TerminationCharacter = "\r"` - Sets the termination character for reading. Default: `\n` (LineFeed or LF)
- `DataChunkSize = 10E3` - Maximum size of one write/read segment. If transferred data is bigger, it is split to more segments. Default: `1E6` bytes
- `OpcTimeout = 10000` - same as driver.utilities.opc_timeout = 10000. Default: `30000ms`
- `VisaTimeout = 5000` - same as driver.utilities.visa_timeout = 5000. Default: `10000ms`

- `ViClearExeMode = Disabled` - viClear() execution mode. Default: `execute_on_all`
- `OpcQueryAfterWrite = True` - same as driver.utilities.opc_query_after_write = True. Default: `False`
- `StbInErrorCheck = False` - if true, the driver checks errors with *STB? If false, it uses SYST:ERR?. Default: `True`
- `ScpiQuotes = double'. - for SCPI commands, you can define how strings are quoted. With single or double quotes. Possible values: single | double | {char}. Default: ``single`
- `LoggingMode = On` - Sets the logging status right from the start. Default: `Off`
- `LoggingName = 'MyDevice'` - Sets the name to represent the session in the log entries. Default: `'resource_name'`
- `LogToGlobalTarget = True` - Sets the logging target to the class-property previously set with RsCMPX_LteMeas.set_global_logging_target() Default: `False`
- `LoggingToConsole = True` - Immediately starts logging to the console. Default: False
- `LoggingToUdp = True` - Immediately starts logging to the UDP port. Default: False
- `LoggingUdpPort = 49200` - UDP port to log to. Default: 49200

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.

classmethod clear_global_logging_relative_timestamp() → None

Clears the global relative timestamp. After this, all the instances using the global relative timestamp continue logging with the absolute timestamps.

close() → None

Closes the active RsCMPX_LteMeas session.

classmethod from_existing_session(*session: object*, *options: str = None*) → RsCMPX_LteMeas

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

Parameters

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

classmethod get_global_logging_relative_timestamp() → datetime

Returns global common relative timestamp for log entries.

classmethod get_global_logging_target()

Returns global common target stream.

get_session_handle() → object

Returns the underlying session handle.

get_total_execution_time() → timedelta

Returns total time spent by the library on communicating with the instrument. This time is always shorter than get_total_time(), since it does not include gaps between the communication. You can reset this counter with reset_time_statistics().

get_total_time() → timedelta

Returns total time spent by the library on communicating with the instrument. This time is always shorter than get_total_time(), since it does not include gaps between the communication. You can reset this counter with reset_time_statistics().

static list_resources(*expression: str = '?*::INSTR'*, *visa_select: 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: '@ivi', '@rs'

reset_time_statistics() → None

Resets all execution and total time counters. Affects the results of get_total_time() and get_total_execution_time()

restore_all_repcaps_to_default() → None

Sets all the Group and Global repcaps to their initial values

classmethod set_global_logging_relative_timestamp(*timestamp: datetime*) → None

Sets global common relative timestamp for log entries. To use it, call the following: io.utilities.logger.set_relative_timestamp_global()

classmethod set_global_logging_relative_timestamp_now() → None

Sets global common relative timestamp for log entries to this moment. To use it, call the following: io.utilities.logger.set_relative_timestamp_global().

classmethod set_global_logging_target(*target*) → None

Sets global common target stream that each instance can use. To use it, call the following: io.utilities.logger.set_logging_target_global(). If an instance uses global logging target, it automatically uses the global relative timestamp (if set). You can set the target to None to invalidate it.

Subgroups

6.1 Configure

class ConfigureCls

Configure commands group definition. 219 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1 LteMeas

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:BAND
CONFigure:LTE:MEASurement<Instance>:SPATh
CONFigure:LTE:MEASurement<Instance>:STYPe
CONFigure:LTE:MEASurement<Instance>:DMODe
CONFigure:LTE:MEASurement<Instance>:FSTRucture
```

class LteMeasCls

LteMeas commands group definition. 219 total commands, 10 Subgroups, 5 group commands

get_band() → Band

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

Selects the operating band (OB) .

INTRO_CMD_HELP: The allowed input range has dependencies:

- FDD UL: OB1 | … | OB28 | OB30 | OB31 | OB65 | OB66 | OB68 | OB70 | … | OB74 | OB85 | OB87 | OB88
- TDD UL: OB33 | … | OB45 | OB48 | OB50 | … | OB53 | OB250

For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:RFSettings:FBINdicator.

return

band: No help available

get_dmode() → Mode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:DMODe
value: enums.Mode = driver.configure.lteMeas.get_dmode()
```

Selects the duplex mode of the LTE signal: FDD or TDD. For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:RFSettings:DMODe.

return
mode: No help available

get_fstructure() → FrameStructure

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:FSTRucture
value: enums.FrameStructure = driver.configure.lteMeas.get_fstructure()
```

Queries the frame structure type of the LTE signal. The value depends on the duplex mode (method RsCMPX_LteMeas.Configure. LteMeas.dmode) .

return
frame_structure: T1: Type 1, FDD signal T2: Type 2, TDD signal

get_spath() → Path

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SPATh
value: enums.Path = driver.configure.lteMeas.get_spath()
```

Selects between a standalone measurement and a measurement with coupling to signaling settings (cell settings of the network configuration) .

return
path: No help available

get_stype() → SignalType

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:STYPe
value: enums.SignalType = driver.configure.lteMeas.get_stype()
```

Selects the type of the measured signal.

return
signal_type: UL: LTE uplink signal SL: V2X sidelink signal

set_band(*band: Band*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:BAND
driver.configure.lteMeas.set_band(band = enums.Band.OB1)
```

Selects the operating band (OB) .
INTRO_CMD_HELP: The allowed input range has dependencies:

- FDD UL: OB1 | … | OB28 | OB30 | OB31 | OB65 | OB66 | OB68 | OB70 | … | OB74 | OB85 | OB87 | OB88
- TDD UL: OB33 | … | OB45 | OB48 | OB50 | … | OB53 | OB250

For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:RFSettings:FBINdicator.

param band
No help available

set_dmode(*mode: Mode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:DMODe
driver.configure.lteMeas.set_dmode(mode = enums.Mode.FDD)
```

Selects the duplex mode of the LTE signal: FDD or TDD. For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:RFSettings:DMODe.

param mode
No help available

set_spath(*path: Path*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SPATh
driver.configure.lteMeas.set_spath(path = enums.Path.NETWork)
```

Selects between a standalone measurement and a measurement with coupling to signaling settings (cell settings of the network configuration) .

param path
No help available

set_stype(*signal_type: SignalType*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:STYPe
driver.configure.lteMeas.set_stype(signal_type = enums.SignalType.SL)
```

Selects the type of the measured signal.

param signal_type
UL: LTE uplink signal SL: V2X sidelink signal

Cloning the Group

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

Subgroups

6.1.1.1 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 12 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.1.1 ChannelBw

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:CBANdwidth:AGGRegated
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_aggregated() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:CBANdwidth:AGGRegated
value: float = driver.configure.lteMeas.carrierAggregation.channelBw.get_
↪aggregated()
```

Queries the width of the aggregated channel bandwidth.

return
ch_bandwidth: No help available

6.1.1.1.2 Frequency

class FrequencyCls

Frequency commands group definition. 3 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.1.2.1 Aggregated

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:LOW
CONFigure:LTE:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:CENTer
CONFigure:LTE:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:HIGH
```

class AggregatedCls

Aggregated commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_center() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:CENTer
value: float = driver.configure.lteMeas.carrierAggregation.frequency.aggregated.
↪get_center()
```

Queries the center frequency of the aggregated bandwidth.

return
frequency_center: No help available

get_high() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:HIGH
value: float = driver.configure.lteMeas.carrierAggregation.frequency.aggregated.
↪get_high()
```

Queries the upper edge of the aggregated bandwidth.

return
frequency_high: No help available

get_low() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:LOW
value: float = driver.configure.lteMeas.carrierAggregation.frequency.aggregated.
↪get_low()
```

Queries the lower edge of the aggregated bandwidth.

return
frequency_low: No help available

6.1.1.1.3 Maping

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:PCC
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing
```

class MapingCls

Maping commands group definition. 3 total commands, 1 Subgroups, 2 group commands

class ValueStruct

Structure for reading output parameters. Fields:

- Cc_1: enums.CarrAggrMaping: No parameter help available
- Cc_2: enums.CarrAggrMaping: No parameter help available
- Cc_3: enums.CarrAggrMaping: No parameter help available
- Cc_4: enums.CarrAggrMaping: No parameter help available

get_pcc() → str

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:PCC
value: str = driver.configure.lteMeas.carrierAggregation.maping.get_pcc()
```

No command help available

return
cc: No help available

get_value() → ValueStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing
value: ValueStruct = driver.configure.lteMeas.carrierAggregation.maping.get_
↪value()
```

No command help available

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.configure.lteMeas.carrierAggregation.maping.clone()
```

Subgroups

6.1.1.1.3.1 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.lteMeas.carrierAggregation.maping.scc.repcap_secondaryCC_get()
driver.configure.lteMeas.carrierAggregation.maping.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:SCC<Carrier>
```

class SccCls

Scc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

get(*secondaryCC=SecondaryCC.Default*) → str

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MAPing:SCC<Carrier>
value: str = driver.configure.lteMeas.carrierAggregation.maping.scc.
↪get(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
cc: No help available

Cloning the Group

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

6.1.1.1.4 Mcarrier

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier:ENHanced
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier
```

class McarrierCls

Mcarrier commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_enhanced() → MeasCarrierEnhanced

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier:ENHanced
value: enums.MeasCarrierEnhanced = driver.configure.lteMeas.carrierAggregation.
↪mcarrier.get_enhanced()
```

Selects a component carrier for single carrier measurements.

return
meas_carrier: No help available

get_value() → MeasCarrierB

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier
value: enums.MeasCarrierB = driver.configure.lteMeas.carrierAggregation.
↪mcarrier.get_value()
```

No command help available

return
meas_carrier: No help available

set_enhanced(*meas_carrier: MeasCarrierEnhanced*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier:ENHanced
driver.configure.lteMeas.carrierAggregation.mcarrier.set_enhanced(meas_carrier␣
↪= enums.MeasCarrierEnhanced.CC1)
```

Selects a component carrier for single carrier measurements.

param meas_carrier
No help available

set_value(*meas_carrier: MeasCarrierB*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MCARrier
driver.configure.lteMeas.carrierAggregation.mcarrier.set_value(meas_carrier =␣
↪enums.MeasCarrierB.PCC)
```

No command help available

param meas_carrier
No help available

6.1.1.1.5 Mode

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE:CSPath
CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE
```

class ModeCls

Mode commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_combined_signal_path() → CarrAggrMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE:CSPath
value: enums.CarrAggrMode = driver.configure.lteMeas.carrierAggregation.mode.
↪get_combined_signal_path()
```

No command help available

return
ca_mode: No help available

get_value() → CarrAggrMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE
value: enums.CarrAggrMode = driver.configure.lteMeas.carrierAggregation.mode.
↪get_value()
```

Selects how many component carriers with intraband contiguous aggregation are measured. For Signal Path = Network, the setting is not configurable.

return
ca_mode: OFF: Only one carrier is measured. INTRaband: two carriers (BW class B & C) ICD: three carriers (BW class D) ICE: four carriers (BW class E)

set_value(*ca_mode: CarrAggrMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation:MODE
driver.configure.lteMeas.carrierAggregation.mode.set_value(ca_mode = enums.
↪CarrAggrMode.ICD)
```

Selects how many component carriers with intraband contiguous aggregation are measured. For Signal Path = Network, the setting is not configurable.

param ca_mode
OFF: Only one carrier is measured. INTRaband: two carriers (BW class B & C) ICD: three carriers (BW class D) ICE: four carriers (BW class E)

6.1.1.1.6 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.lteMeas.carrierAggregation.scc.repcap_secondaryCC_get()
driver.configure.lteMeas.carrierAggregation.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.1.1.1.6.1 AcSpacing

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CAGGregation[:SCC<Nr>]:ACSPacing
```

class AcSpacingCls

AcSpacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*secondaryCC=SecondaryCC.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CAGGregation[:SCC<Nr>]:ACSPacing
driver.configure.lteMeas.carrierAggregation.scc.acSpacing.set(secondaryCC =␣
↪repcap.SecondaryCC.Default)
```

Adjusts the component carrier frequencies, so that the carriers are aggregated contiguously.

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

set_with_opc(*secondaryCC=SecondaryCC.Default*, *opc_timeout_ms: int = -1*) → None

6.1.1.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.lteMeas.cc.repcap_carrierComponent_get()
driver.configure.lteMeas.cc.repcap_carrierComponent_set(repcap.CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.1.1.2.1 ChannelBw

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:CC<Nr>:CBANdwidth
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*carrierComponent=CarrierComponent.Default*) → ChannelBandwidth

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CC<Nr>:CBANdwidth
value: enums.ChannelBandwidth = driver.configure.lteMeas.cc.channelBw.
↪get(carrierComponent = repcap.CarrierComponent.Default)
```

Selects the channel bandwidth of component carrier CC<no>. Without carrier aggregation, you can omit <no>. For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:RFSettings:UL:BWIDth.

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

return
channel_bw: B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

set(*channel_bw: ChannelBandwidth*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:CC<Nr>:CBANdwidth
driver.configure.lteMeas.cc.channelBw.set(channel_bw = enums.ChannelBandwidth.
↪B014, carrierComponent = repcap.CarrierComponent.Default)
```

Selects the channel bandwidth of component carrier CC<no>. Without carrier aggregation, you can omit <no>. For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:RFSettings:UL:BWIDth.

param channel_bw
B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

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

6.1.1.3 Emtc

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:EMTC:ENABle
CONFigure:LTE:MEASurement<instance>:EMTC:MB<number>
CONFigure:LTE:MEASurement<Instance>:EMTC:NBANd
```

class EmtcCls

Emtc commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_enable() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:ENABle
value: bool = driver.configure.lteMeas.emtc.get_enable()
```

Enables or disables eMTC. For Signal Path = Network, the setting is not configurable.

return
enable: No help available

get_mb() → bool

```
# SCPI: CONFigure:LTE:MEASurement<instance>:EMTC:MB<number>
value: bool = driver.configure.lteMeas.emtc.get_mb()
```

Selects the maximum eMTC bandwidth.

return
enable: OFF: Max bandwidth 1.4 MHz ON: Max bandwidth 5 MHz

get_nband() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:NBANd
value: int = driver.configure.lteMeas.emtc.get_nband()
```

Selects the narrowband used for eMTC.

return
number: The maximum depends on the channel BW, see 'RB allocation, narrowbands and widebands for eMTC'.

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:ENABle
driver.configure.lteMeas.emtc.set_enable(enable = False)
```

Enables or disables eMTC. For Signal Path = Network, the setting is not configurable.

param enable
No help available

set_mb(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<instance>:EMTC:MB<number>
driver.configure.lteMeas.emtc.set_mb(enable = False)
```

Selects the maximum eMTC bandwidth.

param enable
OFF: Max bandwidth 1.4 MHz ON: Max bandwidth 5 MHz

set_nband(*number: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:EMTC:NBANd
driver.configure.lteMeas.emtc.set_nband(number = 1)
```

Selects the narrowband used for eMTC.

param number
The maximum depends on the channel BW, see 'RB allocation, narrowbands and wideband for eMTC'.

6.1.1.4 MultiEval

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:TOUT
CONFigure:LTE:MEASurement<Instance>:MEValuation:MMODe
CONFigure:LTE:MEASurement<Instance>:MEValuation:REPetition
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCONdition
CONFigure:LTE:MEASurement<Instance>:MEValuation:ULDL
CONFigure:LTE:MEASurement<Instance>:MEValuation:SSUBframe
CONFigure:LTE:MEASurement<Instance>:MEValuation:MOEXception
CONFigure:LTE:MEASurement<Instance>:MEValuation:CPRefix
CONFigure:LTE:MEASurement<Instance>:MEValuation:CTYPe
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCTYpe
CONFigure:LTE:MEASurement<Instance>:MEValuation:PSEarch
CONFigure:LTE:MEASurement<Instance>:MEValuation:PFORmat
CONFigure:LTE:MEASurement<Instance>:MEValuation:NVFilter
CONFigure:LTE:MEASurement<Instance>:MEValuation:ORVFilter
CONFigure:LTE:MEASurement<Instance>:MEValuation:CTVFilter
CONFigure:LTE:MEASurement<Instance>:MEValuation:DSSPusch
CONFigure:LTE:MEASurement<Instance>:MEValuation:GHOPping
CONFigure:LTE:MEASurement<Instance>:MEValuation:MSLot
CONFigure:LTE:MEASurement<Instance>:MEValuation:VIEW
```

class MultiEvalCls

MultiEval commands group definition. 141 total commands, 17 Subgroups, 19 group commands

get_cprefix() → CyclicPrefix

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CPRefix
value: enums.CyclicPrefix = driver.configure.lteMeas.multiEval.get_cprefix()
```

Selects the type of cyclic prefix of the LTE signal. For Signal Path = Network, the setting is not configurable.

return
cyclic_prefix: No help available

get_ctv_filter() → ChannelTypeVewFilter

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTVFilter
value: enums.ChannelTypeVewFilter = driver.configure.lteMeas.multiEval.get_ctv_
↪filter()
```

Specifies, enables or disables the channel type view filter. If the filter is active, only slots with detected channel type PUSCH or PUCCH are measured.

return
channel_type: PUSCh: measure only PUSCH PUCCh: measure only PUCCH ON: enable the filter OFF: disable the filter

get_ctype() → ChannelTypeDetection

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTYPe
value: enums.ChannelTypeDetection = driver.configure.lteMeas.multiEval.get_
↪ctype()
```

Configures the channel type detection for uplink measurements.

return
channel_type: Automatic detection of channel type or manual selection

get_dss_pusch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:DSSPusch
value: int = driver.configure.lteMeas.multiEval.get_dss_pusch()
```

Specifies the delta sequence shift value (ss) used to calculate the sequence shift pattern for PUSCH.

return
delta_seq_sh_pusch: No help available

get_ghopping() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:GHOPping
value: bool = driver.configure.lteMeas.multiEval.get_ghopping()
```

Specifies whether group hopping is used or not. For Signal Path = Network, the setting is not configurable.

return
value: No help available

get_mmode() → MeasurementMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MMODe
value: enums.MeasurementMode = driver.configure.lteMeas.multiEval.get_mmode()
```

Selects the measurement mode.

return
measurement_mode: NORMal: normal mode TMODe: TPC mode MELMode: multi-evaluation list mode

get_mo_exception() → bool

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

Specifies whether measurement results identified as faulty or inaccurate are rejected.

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

get_mslot() → MeasureSlot

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSLot
value: enums.MeasureSlot = driver.configure.lteMeas.multiEval.get_mslot()
```

Selects which slots of the Measure Subframe are measured.

return
measure_slot: MS0: slot number 0 only MS1: slot number 1 only ALL: both slots

get_nvfilter() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NVFilter
value: int or bool = driver.configure.lteMeas.multiEval.get_nvfilter()
```

Specifies, enables or disables the number of resource blocks (NRB) view filter. If the filter is active, only slots with a matching number of allocated resource blocks are measured. Within the indicated input range, only specific numbers are allowed as defined in 3GPP TS 36.211. For details, see 'Resources in time and frequency domain'.

return
nrb_view_filter: (integer or boolean) Number of allocated resource blocks

get_orv_filter() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ORVFilter
value: int or bool = driver.configure.lteMeas.multiEval.get_orv_filter()
```

Specifies, enables or disables the RB offset view filter. If the filter is active, only slots with a matching number of RB offset are measured. The indicated input range applies to a 20-MHz channel bandwidth. The maximum value depends on the bandwidth (maximum number of RBs minus one) .

return
offset_rb: (integer or boolean) Offset of the first allocated RB

get_peak_search() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PSEarch
value: bool = driver.configure.lteMeas.multiEval.get_peak_search()
```

No command help available

return
pucch_search: No help available

get_pformat() → PucchFormat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PFORmat
value: enums.PucchFormat = driver.configure.lteMeas.multiEval.get_pformat()
```

Specifies the PUCCH format (only relevant for signals containing a PUCCH) . The formats are defined in 3GPP TS 36.211.

return
pucch_format: No help available

get_repetition() → Repeat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:REPetition
value: enums.Repeat = driver.configure.lteMeas.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: Single-shot measurement CONTinuous: Continuous measurement

get_sch_type() → SidelinkChannelType

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCTYpe
value: enums.SidelinkChannelType = driver.configure.lteMeas.multiEval.get_sch_
↪type()
```

Configures the channel type for modulation results of sidelink measurements.

return
channel_type: No help available

get_scondition() → StopCondition

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCONdition
value: enums.StopCondition = driver.configure.lteMeas.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: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

get_ssubframe() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SSUBframe
value: int = driver.configure.lteMeas.multiEval.get_ssubframe()
```

Selects a special subframe configuration, defining the inner structure of special subframes. This parameter is only relevant for frame structure Type 2 (method RsCMPX_LteMeas.Configure.LteMeas.fstructure) . The special subframe configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

return
special_subframe: No help available

get_timeout() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TOUT
value: float = driver.configure.lteMeas.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. 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: No help available

get_ul_dl() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ULDL
value: int = driver.configure.lteMeas.multiEval.get_ul_dl()
```

Selects an UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame. This parameter is only relevant for frame structure Type 2 (method RsCMPX_LteMeas.Configure.LteMeas.fstructure) . The UL-DL configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

return
uplink_downlink: No help available

get_view() → ViewMev

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:VIEW
value: enums.ViewMev = driver.configure.lteMeas.multiEval.get_view()
```

No command help available

return
view: No help available

set_cprefix(*cyclic_prefix: CyclicPrefix*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CPRefix
driver.configure.lteMeas.multiEval.set_cprefix(cyclic_prefix = enums.
↪CyclicPrefix.EXTended)
```

Selects the type of cyclic prefix of the LTE signal. For Signal Path = Network, the setting is not configurable.

param cyclic_prefix
No help available

set_ctv_filter(*channel_type: ChannelTypeVewFilter*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTVFilter
driver.configure.lteMeas.multiEval.set_ctv_filter(channel_type = enums.
↪ChannelTypeVewFilter.OFF)
```

Specifies, enables or disables the channel type view filter. If the filter is active, only slots with detected channel type PUSCH or PUCCH are measured.

param channel_type
PUSCh: measure only PUSCH PUCCh: measure only PUCCH ON: enable the filter
OFF: disable the filter

set_ctype(*channel_type: ChannelTypeDetection*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CTYPe
driver.configure.lteMeas.multiEval.set_ctype(channel_type = enums.
↪ChannelTypeDetection.AUTO)
```

Configures the channel type detection for uplink measurements.

param channel_type
Automatic detection of channel type or manual selection

set_dss_pusch(*delta_seq_sh_pusch: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:DSSPusch
driver.configure.lteMeas.multiEval.set_dss_pusch(delta_seq_sh_pusch = 1)
```

Specifies the delta sequence shift value (ss) used to calculate the sequence shift pattern for PUSCH.

param delta_seq_sh_pusch
No help available

set_ghopping(*value: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:GHOPping
driver.configure.lteMeas.multiEval.set_ghopping(value = False)
```

Specifies whether group hopping is used or not. For Signal Path = Network, the setting is not configurable.

param value
No help available

set_mmode(*measurement_mode: MeasurementMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MMODe
driver.configure.lteMeas.multiEval.set_mmode(measurement_mode = enums.
↪MeasurementMode.MELMode)
```

Selects the measurement mode.

param measurement_mode
NORMal: normal mode TMODe: TPC mode MELMode: multi-evaluation list mode

set_mo_exception(*meas_on_exception: bool*) → None

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

Specifies whether measurement results identified as faulty or inaccurate are rejected.

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

set_mslot(*measure_slot: MeasureSlot*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSLot
driver.configure.lteMeas.multiEval.set_mslot(measure_slot = enums.MeasureSlot.
↪ALL)
```

Selects which slots of the Measure Subframe are measured.

param measure_slot
MS0: slot number 0 only MS1: slot number 1 only ALL: both slots

set_nvfilter(*nrb_view_filter: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NVFilter
driver.configure.lteMeas.multiEval.set_nvfilter(nrb_view_filter = 1)
```

Specifies, enables or disables the number of resource blocks (NRB) view filter. If the filter is active, only slots with a matching number of allocated resource blocks are measured. Within the indicated input range, only specific numbers are allowed as defined in 3GPP TS 36.211. For details, see 'Resources in time and frequency domain'.

param nrb_view_filter
(integer or boolean) Number of allocated resource blocks

set_orv_filter(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ORVFilter
driver.configure.lteMeas.multiEval.set_orv_filter(offset_rb = 1)
```

Specifies, enables or disables the RB offset view filter. If the filter is active, only slots with a matching number of RB offset are measured. The indicated input range applies to a 20-MHz channel bandwidth. The maximum value depends on the bandwidth (maximum number of RBs minus one) .

param offset_rb
(integer or boolean) Offset of the first allocated RB

set_peak_search(*pucch_search: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PSEarch
driver.configure.lteMeas.multiEval.set_peak_search(pucch_search = False)
```

No command help available

param pucch_search
No help available

set_pformat(*pucch_format: PucchFormat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PFORmat
driver.configure.lteMeas.multiEval.set_pformat(pucch_format = enums.PucchFormat.
↪F1)
```

Specifies the PUCCH format (only relevant for signals containing a PUCCH) . The formats are defined in 3GPP TS 36.211.

param pucch_format
No help available

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:REPetition
driver.configure.lteMeas.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: Single-shot measurement CONTinuous: Continuous measurement

set_sch_type(*channel_type: SidelinkChannelType*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCTYpe
driver.configure.lteMeas.multiEval.set_sch_type(channel_type = enums.
↪SidelinkChannelType.PSBCh)
```

Configures the channel type for modulation results of sidelink measurements.

param channel_type
No help available

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCONdition
driver.configure.lteMeas.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: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

set_ssubframe(*special_subframe: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SSUBframe
driver.configure.lteMeas.multiEval.set_ssubframe(special_subframe = 1)
```

Selects a special subframe configuration, defining the inner structure of special subframes. This parameter is only relevant for frame structure Type 2 (method RsCMPX_LteMeas.Configure.LteMeas.fstructure) . The special subframe configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

param special_subframe
No help available

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TOUT
driver.configure.lteMeas.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. 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
No help available

set_ul_dl(*uplink_downlink: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:ULDL
driver.configure.lteMeas.multiEval.set_ul_dl(uplink_downlink = 1)
```

Selects an UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame. This parameter is only relevant for frame structure Type 2 (method RsCMPX_LteMeas.Configure.LteMeas.fstructure) . The UL-DL configurations are defined in 3GPP TS 36.211, chapter 4, 'Frame Structure'.

param uplink_downlink
No help available

set_view(*view: ViewMev*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:VIEW
driver.configure.lteMeas.multiEval.set_view(view = enums.ViewMev.ACLR)
```

No command help available

param view
No help available

Cloning the Group

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

Subgroups

6.1.1.4.1 Bler

class BlerCls

Bler commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.1.1 Sframes

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:BLER:SFRames
```

class SframesCls

Sframes commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SframesStruct

Response structure. Fields:

- Sub_Frames: int: No parameter help available
- Sched_Subfr_Per_Fr: int: No parameter help available

get() → SframesStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:BLER:SFRames
value: SframesStruct = driver.configure.lteMeas.multiEval.bler.sframes.get()
```

No command help available

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

set(*sub_frames: int*, *sched_subfr_per_fr: int = None*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:BLER:SFRames
driver.configure.lteMeas.multiEval.bler.sframes.set(sub_frames = 1, sched_subfr_
↪per_fr = 1)
```

No command help available

param sub_frames
No help available

param sched_subfr_per_fr
No help available

6.1.1.4.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.lteMeas.multiEval.cc.repcap_carrierComponent_get()
driver.configure.lteMeas.multiEval.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.1.1.4.2.1 PlcId

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:CC<Nr>:PLCid
```

class PlcIdCls

PlcId commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*carrierComponent=CarrierComponent.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CC<Nr>:PLCid
value: int = driver.configure.lteMeas.multiEval.cc.plcId.get(carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Specifies the physical layer cell ID of component carrier CC<no>. Without carrier aggregation, you can omit <no>. For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:PCID.

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

return
phs_layer_cell_id: No help available

set(*phs_layer_cell_id: int*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:CC<Nr>:PLCid
driver.configure.lteMeas.multiEval.cc.plcId.set(phs_layer_cell_id = 1,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Specifies the physical layer cell ID of component carrier CC<no>. Without carrier aggregation, you can omit <no>. For Signal Path = Network, use [CONFigure:]SIGNaling:LTE:CELL:PCID.

param phs_layer_cell_id
No help available

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

6.1.1.4.3 Limit

class LimitCls

Limit commands group definition. 40 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.1 Aclr

class AclrCls

Aclr commands group definition. 8 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.2 Eutra

class EutraCls

Eutra commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.3 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.4 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.
↪channelBw1st.clone()
```

Subgroups

6.1.1.4.3.5 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth
↪<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.lteMeas.multiEval.limit.aclr.eutra.
↪carrierAggregation.channelBw1st.channelBw2nd.get(firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth100:CBANdwidth50.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.
↪channelBw1st.channelBw2nd.set(relative_level = 1.0, absolute_level = 1.0,␣
↪firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw = repcap.
↪SecondChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth100:CBANdwidth50.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.
↪channelBw1st.channelBw2nd.clone()
```

Subgroups

6.1.1.4.3.6 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth
↪<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

class ChannelBw3rdStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → ChannelBw3rdStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
↪:CBANdwidth<Band3>
value: ChannelBw3rdStruct = driver.configure.lteMeas.multiEval.limit.aclr.eutra.
↪carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(firstChannelBw␣
↪= repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

return
structure: for return value, see the help for ChannelBw3rdStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
↪:CBANdwidth<Band3>
driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.set(relative_level = 1.0, absolute_
↪level = 1.0, firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw␣
↪= repcap.SecondChannelBw.Default, thirdChannelBw = repcap.ThirdChannelBw.
↪Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.clone()
```

6.1.1.4.3.7 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:OCOMbination
```

class OcombinationCls

Ocombination commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class OcombinationStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get() → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.lteMeas.multiEval.limit.aclr.eutra.
↪carrierAggregation.ocombination.get()
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

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

set(*relative_level: float*, *absolute_level: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CAGGregation:OCOMbination
driver.configure.lteMeas.multiEval.limit.aclr.eutra.carrierAggregation.
↪ocombination.set(relative_level = 1.0, absolute_level = 1.0)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

6.1.1.4.3.8 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.eutra.channelBw.repcap_channelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.eutra.channelBw.repcap_channelBw_
↪set(repcap.ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:EUTRa:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get(*channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.lteMeas.multiEval.limit.aclr.eutra.
↪channelBw.get(channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*relative_level: float*, *absolute_level: float*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:EUTRa:CBANdwidth<Band>
driver.configure.lteMeas.multiEval.limit.aclr.eutra.channelBw.set(relative_
↪level = 1.0, absolute_level = 1.0, channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent E-UTRA channel. The settings are defined separately for each channel bandwidth.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.1.4.3.9 Utra<UtraAdjChannel>

RepCap Settings

```
# Range: Ch1 .. Ch2
rc = driver.configure.lteMeas.multiEval.limit.aclr.utra.repcap_utraAdjChannel_get()
driver.configure.lteMeas.multiEval.limit.aclr.utra.repcap_utraAdjChannel_set(repcap.
↪UtraAdjChannel.Ch1)
```

class UtraCls

Utra commands group definition. 4 total commands, 2 Subgroups, 0 group commands Repeated Capability: UtraAdjChannel, default value after init: UtraAdjChannel.Ch1

Cloning the Group

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

Subgroups

6.1.1.4.3.10 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.11 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.
↪channelBw1st.clone()
```

Subgroups

6.1.1.4.3.12 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get(*utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.lteMeas.multiEval.limit.aclr.utra.
↪carrierAggregation.channelBw1st.channelBw2nd.get(utraAdjChannel = repcap.
↪UtraAdjChannel.Default, firstChannelBw = repcap.FirstChannelBw.Default,␣
↪secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for two

aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use … :CBANdwidth100:CBANdwidth50.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.
↪channelBw1st.channelBw2nd.set(relative_level = 1.0, absolute_level = 1.0,␣
↪utraAdjChannel = repcap.UtraAdjChannel.Default, firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use … :CBANdwidth100:CBANdwidth50.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.
↪channelBw1st.channelBw2nd.clone()
```

Subgroups

6.1.1.4.3.13 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

class ChannelBw3rdStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get(*utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → ChannelBw3rdStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
value: ChannelBw3rdStruct = driver.configure.lteMeas.multiEval.limit.aclr.utra.
↪carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(utraAdjChannel␣
↪= repcap.UtraAdjChannel.Default, firstChannelBw = repcap.FirstChannelBw.
↪Default, secondChannelBw = repcap.SecondChannelBw.Default, thirdChannelBw =␣
↪repcap.ThirdChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use … :CBANdwidth200:CBANdwidth150:CBANdwidth100.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

return
structure: for return value, see the help for ChannelBw3rdStruct structure arguments.

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.set(relative_level = 1.0, absolute_
↪level = 1.0, utraAdjChannel = repcap.UtraAdjChannel.Default, firstChannelBw =␣
↪repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use … :CBANdwidth200:CBANdwidth150:CBANdwidth100.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.clone()
```

6.1.1.4.3.14 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:OCOMbination
```

class OcombinationCls

Ocombination commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class OcombinationStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get(*utraAdjChannel=UtraAdjChannel.Default*) → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.lteMeas.multiEval.limit.aclr.utra.
↪carrierAggregation.ocombination.get(utraAdjChannel = repcap.UtraAdjChannel.
↪Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

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

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CAGGregation:OCOMbination
driver.configure.lteMeas.multiEval.limit.aclr.utra.carrierAggregation.
↪ocombination.set(relative_level = 1.0, absolute_level = 1.0, utraAdjChannel =␣
↪repcap.UtraAdjChannel.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

6.1.1.4.3.15 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.aclr.utra.channelBw.repcap_channelBw_get()
driver.configure.lteMeas.multiEval.limit.aclr.utra.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Relative_Level: float or bool: No parameter help available
- Absolute_Level: float or bool: No parameter help available

get(*utraAdjChannel=UtraAdjChannel.Default*, *channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.lteMeas.multiEval.limit.aclr.utra.
↪channelBw.get(utraAdjChannel = repcap.UtraAdjChannel.Default, channelBw =␣
↪repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth.

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*relative_level: float*, *absolute_level: float*, *utraAdjChannel=UtraAdjChannel.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<nr>
↪:CBANdwidth<Band>
driver.configure.lteMeas.multiEval.limit.aclr.utra.channelBw.set(relative_level␣
↪= 1.0, absolute_level = 1.0, utraAdjChannel = repcap.UtraAdjChannel.Default,␣
↪channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on <no>. The settings are defined separately for each channel bandwidth.

param relative_level
(float or boolean) No help available

param absolute_level
(float or boolean) No help available

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.1.4.3.16 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.17 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.pdynamics.channelBw.repcap_channelBw_get()
driver.configure.lteMeas.multiEval.limit.pdynamics.channelBw.repcap_channelBw_set(repcap.
↪ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- On_Power_Upper: float: Upper limit for the 'ON power'
- On_Power_Lower: float: Lower limit for the 'ON power'
- Off_Power_Upper: float: Upper limit for the 'OFF power' and the 'SRS OFF' power

get(*channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.lteMeas.multiEval.limit.pdynamics.
↪channelBw.get(channelBw = repcap.ChannelBw.Default)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement. Separate limits can be defined for each channel bandwidth.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*enable: bool*, *on_power_upper: float*, *on_power_lower: float*, *off_power_upper: float*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:CBANdwidth<Band>
driver.configure.lteMeas.multiEval.limit.pdynamics.channelBw.set(enable = False,
↪ on_power_upper = 1.0, on_power_lower = 1.0, off_power_upper = 1.0, channelBw␣
↪= repcap.ChannelBw.Default)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement. Separate limits can be defined for each channel bandwidth.

param enable
OFF: disables the limit check ON: enables the limit check

param on_power_upper
Upper limit for the 'ON power'

param on_power_lower
Lower limit for the 'ON power'

param off_power_upper
Upper limit for the 'OFF power' and the 'SRS OFF' power

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.1.4.3.18 Qam<QAMmodOrder>

RepCap Settings

```
# Range: Qam16 .. Qam256
rc = driver.configure.lteMeas.multiEval.limit.qam.repcap_qAMmodOrder_get()
driver.configure.lteMeas.multiEval.limit.qam.repcap_qAMmodOrder_set(repcap.QAMmodOrder.
↪Qam16)
```

class QamCls

Qam commands group definition. 9 total commands, 8 Subgroups, 0 group commands Repeated Capability: QAMmodOrder, default value after init: QAMmodOrder.Qam16

Cloning the Group

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

Subgroups

6.1.1.4.3.19 EsFlatness

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:ESFLatness
```

class EsFlatnessCls

EsFlatness commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EsFlatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- Range_1: float: Upper limit for max(range 1) - min(range 1)
- Range_2: float: Upper limit for max(range 2) - min(range 2)
- Max_1_Min_2: float: Upper limit for max(range 1) - min(range 2)
- Max_2_Min_1: float: Upper limit for max(range 2) - min(range 1)
- Edge_Frequency: float: Frequency band edge distance of border between range 1 and range 2

get(*qAMmodOrder=QAMmodOrder.Default*) → EsFlatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:ESFLatness
value: EsFlatnessStruct = driver.configure.lteMeas.multiEval.limit.qam.
↪esFlatness.get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines limits for the equalizer spectrum flatness, for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*structure: EsFlatnessStruct*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:ESFLatness
structure = driver.configure.lteMeas.multiEval.limit.qam.esFlatness.
↪EsFlatnessStruct()
structure.Enable: bool = False
structure.Range_1: float = 1.0
structure.Range_2: float = 1.0
structure.Max_1_Min_2: float = 1.0
structure.Max_2_Min_1: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.lteMeas.multiEval.limit.qam.esFlatness.set(structure,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines limits for the equalizer spectrum flatness, for QAM modulations.

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

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.20 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:EVMagnitude
```

class EvMagnitudeCls

EvMagnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EvMagnitudeStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get(*qAMmodOrder=QAMmodOrder.Default*) → EvMagnitudeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:EVMagnitude
value: EvMagnitudeStruct = driver.configure.lteMeas.multiEval.limit.qam.
↪evMagnitude.get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) , for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:EVMagnitude
driver.configure.lteMeas.multiEval.limit.qam.evMagnitude.set(rms = 1.0, peak =␣
↪1.0, qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) , for QAM modulations.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.21 FreqError

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:FERRor
```

class FreqErrorCls

FreqError commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*qAMmodOrder=QAMmodOrder.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:FERRor
value: float or bool = driver.configure.lteMeas.multiEval.limit.qam.freqError.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines an upper limit for the carrier frequency error for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

return
frequency_error: (float or boolean) No help available

set(*frequency_error: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:FERRor
driver.configure.lteMeas.multiEval.limit.qam.freqError.set(frequency_error = 1.
↪0, qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines an upper limit for the carrier frequency error for QAM modulations.

param frequency_error
(float or boolean) No help available

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.22 Ibe

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE
```

class IbeCls

Ibe commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class IbeStruct

Response structure. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- Minimum: float: No parameter help available

- Evm: float: No parameter help available
- Rb_Power: float: No parameter help available
- Iq_Image: float: No parameter help available

get(*qAMmodOrder=QAMmodOrder.Default*) → IbeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE
value: IbeStruct = driver.configure.lteMeas.multiEval.limit.qam.ibe.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines parameters used for calculation of an upper limit for the in-band emission, for QAM modulations, see 'In-band emissions limits'.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*enable: bool*, *minimum: float*, *evm: float*, *rb_power: float*, *iq_image: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE
driver.configure.lteMeas.multiEval.limit.qam.ibe.set(enable = False, minimum =␣
↪1.0, evm = 1.0, rb_power = 1.0, iq_image = 1.0, qAMmodOrder = repcap.
↪QAMmodOrder.Default)
```

Defines parameters used for calculation of an upper limit for the in-band emission, for QAM modulations, see 'In-band emissions limits'.

param enable
OFF: disables the limit check ON: enables the limit check

param minimum
No help available

param evm
No help available

param rb_power
No help available

param iq_image
No help available

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

Cloning the Group

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

Subgroups

6.1.1.4.3.23 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IBE:IQOFfset
```

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class IqOffsetStruct

Response structure. Fields:

- Offset_1: float: Offset for high TX power range
- Offset_2: float: Offset for intermediate TX power range
- Offset_3: float: Offset for low TX power range

get(*qAMmodOrder=QAMmodOrder.Default*) → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IBE:IQOFfset
value: IqOffsetStruct = driver.configure.lteMeas.multiEval.limit.qam.ibe.
↪iqOffset.get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission, for QAM modulations. Three different values can be set for three TX power ranges, see 'In-band emissions limits'.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*offset_1: float*, *offset_2: float*, *offset_3: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IBE:IQOFfset
driver.configure.lteMeas.multiEval.limit.qam.ibe.iqOffset.set(offset_1 = 1.0,␣
↪offset_2 = 1.0, offset_3 = 1.0, qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission, for QAM modulations. Three different values can be set for three TX power ranges, see 'In-band emissions limits'.

param offset_1
Offset for high TX power range

param offset_2
Offset for intermediate TX power range

param offset_3
Offset for low TX power range

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.24 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:IQOFfset
```

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class IqOffsetStruct

Response structure. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- Offset_1: float: I/Q origin offset limit for high TX power range
- Offset_2: float: I/Q origin offset limit for intermediate TX power range
- Offset_3: float: I/Q origin offset limit for low TX power range

get(*qAMmodOrder=QAMmodOrder.Default*) → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQOFfset
value: IqOffsetStruct = driver.configure.lteMeas.multiEval.limit.qam.iqOffset.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the I/Q origin offset for QAM modulations. Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*enable: bool*, *offset_1: float*, *offset_2: float*, *offset_3: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:IQOFfset
driver.configure.lteMeas.multiEval.limit.qam.iqOffset.set(enable = False,␣
↪offset_1 = 1.0, offset_2 = 1.0, offset_3 = 1.0, qAMmodOrder = repcap.
↪QAMmodOrder.Default)
```

Defines upper limits for the I/Q origin offset for QAM modulations. Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

param enable
OFF: disables the limit check ON: enables the limit check

param offset_1
I/Q origin offset limit for high TX power range

param offset_2
I/Q origin offset limit for intermediate TX power range

param offset_3
I/Q origin offset limit for low TX power range

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.25 Merror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:MERRor
```

`class MerrorCls`

Merror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

`class MerrorStruct`

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

`get`(*qAMmodOrder=QAMmodOrder.Default*) → MerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:MERRor
value: MerrorStruct = driver.configure.lteMeas.multiEval.limit.qam.merror.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the magnitude error, for QAM modulations.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

`set`(*rms: float*, *peak: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:MERRor
driver.configure.lteMeas.multiEval.limit.qam.merror.set(rms = 1.0, peak = 1.0,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines upper limits for the RMS and peak values of the magnitude error, for QAM modulations.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.26 Perror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:PERRor
```

class PerrorCls

Perror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PerrorStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get(*qAMmodOrder=QAMmodOrder.Default*) → PerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PERRor
value: PerrorStruct = driver.configure.lteMeas.multiEval.limit.qam.perror.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines symmetric limits for the RMS and peak values of the phase error, for QAM modulations. The limit check fails if the absolute value of the measured phase error exceeds the specified limit.

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:PERRor
driver.configure.lteMeas.multiEval.limit.qam.perror.set(rms = 1.0, peak = 1.0,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

Defines symmetric limits for the RMS and peak values of the phase error, for QAM modulations. The limit check fails if the absolute value of the measured phase error exceeds the specified limit.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.27 Sflatness

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>:SFLatness
```

class SflatnessCls

Sflatness commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SflatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: No parameter help available
- Lower: float: No parameter help available
- Upper: float: No parameter help available
- Edge_Lower: float: No parameter help available
- Edge_Upper: float: No parameter help available
- Edge_Frequency: float: No parameter help available

get(*qAMmodOrder=QAMmodOrder.Default*) → SflatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SFLatness
value: SflatnessStruct = driver.configure.lteMeas.multiEval.limit.qam.sflatness.
↪get(qAMmodOrder = repcap.QAMmodOrder.Default)
```

No command help available

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

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

set(*structure: SflatnessStruct*, *qAMmodOrder=QAMmodOrder.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QAM<ModOrder>
↪:SFLatness
structure = driver.configure.lteMeas.multiEval.limit.qam.sflatness.
↪SflatnessStruct()
structure.Enable: bool = False
structure.Lower: float = 1.0
structure.Upper: float = 1.0
structure.Edge_Lower: float = 1.0
structure.Edge_Upper: float = 1.0
structure.Edge_Frequency: float = 1.0
```

(continues on next page)

(continued from previous page)

```
driver.configure.lteMeas.multiEval.limit.qam.sflatness.set(structure,␣
↪qAMmodOrder = repcap.QAMmodOrder.Default)
```

No command help available

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

param qAMmodOrder
optional repeated capability selector. Default value: Qam16 (settable in the interface 'Qam')

6.1.1.4.3.28 Qpsk

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:SFLatness
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
```

class QpskCls

Qpsk commands group definition. 9 total commands, 5 Subgroups, 3 group commands

class EsFlatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- Range_1: float: Upper limit for max(range 1) - min(range 1)
- Range_2: float: Upper limit for max(range 2) - min(range 2)
- Max_1_Min_2: float: Upper limit for max(range 1) - min(range 2)
- Max_2_Min_1: float: Upper limit for max(range 2) - min(range 1)
- Edge_Frequency: float: Frequency band edge distance of border between range 1 and range 2

class SflatnessStruct

Structure for setting input parameters. Fields:

- Enable: bool: No parameter help available
- Lower: float: No parameter help available
- Upper: float: No parameter help available
- Edge_Lower: float: No parameter help available
- Edge_Upper: float: No parameter help available
- Edge_Frequency: float: No parameter help available

get_es_flatness() → EsFlatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
value: EsFlatnessStruct = driver.configure.lteMeas.multiEval.limit.qpsk.get_es_
↪flatness()
```

Defines limits for the equalizer spectrum flatness (QPSK modulation) .

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

get_freq_error() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
value: float or bool = driver.configure.lteMeas.multiEval.limit.qpsk.get_freq_
↪error()
```

Defines an upper limit for the carrier frequency error (QPSK modulation) .

return
frequency_error: (float or boolean) No help available

get_sflatness() → SflatnessStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:SFLatness
value: SflatnessStruct = driver.configure.lteMeas.multiEval.limit.qpsk.get_
↪sflatness()
```

No command help available

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

set_es_flatness(*value: EsFlatnessStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
structure = driver.configure.lteMeas.multiEval.limit.qpsk.EsFlatnessStruct()
structure.Enable: bool = False
structure.Range_1: float = 1.0
structure.Range_2: float = 1.0
structure.Max_1_Min_2: float = 1.0
structure.Max_2_Min_1: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.lteMeas.multiEval.limit.qpsk.set_es_flatness(value = structure)
```

Defines limits for the equalizer spectrum flatness (QPSK modulation) .

param value
see the help for EsFlatnessStruct structure arguments.

set_freq_error(*frequency_error: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
driver.configure.lteMeas.multiEval.limit.qpsk.set_freq_error(frequency_error =␣
↪1.0)
```

Defines an upper limit for the carrier frequency error (QPSK modulation) .

param frequency_error
(float or boolean) No help available

set_sflatness(*value: SflatnessStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:SFLatness
structure = driver.configure.lteMeas.multiEval.limit.qpsk.SflatnessStruct()
structure.Enable: bool = False
structure.Lower: float = 1.0
structure.Upper: float = 1.0
structure.Edge_Lower: float = 1.0
structure.Edge_Upper: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.lteMeas.multiEval.limit.qpsk.set_sflatness(value = structure)
```

No command help available

param value
see the help for SflatnessStruct structure arguments.

Cloning the Group

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

Subgroups

6.1.1.4.3.29 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
```

class EvMagnitudeCls

EvMagnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EvMagnitudeStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get() → EvMagnitudeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
value: EvMagnitudeStruct = driver.configure.lteMeas.multiEval.limit.qpsk.
↪evMagnitude.get()
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for QPSK.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
driver.configure.lteMeas.multiEval.limit.qpsk.evMagnitude.set(rms = 1.0, peak =␣
↪1.0)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for QPSK.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.4.3.30 Ibe

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
```

class IbeCls

Ibe commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class IbeStruct

Response structure. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- Minimum: float: No parameter help available
- Evm: float: No parameter help available
- Rb_Power: float: No parameter help available
- Iq_Image: float: No parameter help available

get() → IbeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
value: IbeStruct = driver.configure.lteMeas.multiEval.limit.qpsk.ibe.get()
```

Defines parameters used for calculation of an upper limit for the in-band emission (QPSK modulation) , see 'In-band emissions limits'.

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

set(*enable: bool*, *minimum: float*, *evm: float*, *rb_power: float*, *iq_image: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
driver.configure.lteMeas.multiEval.limit.qpsk.ibe.set(enable = False, minimum =␣
↪1.0, evm = 1.0, rb_power = 1.0, iq_image = 1.0)
```

Defines parameters used for calculation of an upper limit for the in-band emission (QPSK modulation) , see 'In-band emissions limits'.

param enable
OFF: disables the limit check ON: enables the limit check

param minimum
No help available

param evm
No help available

param rb_power
No help available

param iq_image
No help available

Cloning the Group

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

Subgroups

6.1.1.4.3.31 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE:IQOFfset
```

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class IqOffsetStruct

Response structure. Fields:

- Offset_1: float: Offset for high TX power range
- Offset_2: float: Offset for intermediate TX power range
- Offset_3: float: Offset for low TX power range

get() → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE:IQOFfset
value: IqOffsetStruct = driver.configure.lteMeas.multiEval.limit.qpsk.ibe.
↪iqOffset.get()
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission (QPSK modulation) . Three different values can be set for three TX power ranges, see 'In-band emissions limits'.

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

set(*offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE:IQOFfset
driver.configure.lteMeas.multiEval.limit.qpsk.ibe.iqOffset.set(offset_1 = 1.0,␣
↪offset_2 = 1.0, offset_3 = 1.0)
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission (QPSK modulation) . Three different values can be set for three TX power ranges, see 'In-band emissions limits'.

param offset_1
Offset for high TX power range

param offset_2
Offset for intermediate TX power range

param offset_3
Offset for low TX power range

6.1.1.4.3.32 IqOffset

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IQOFfset
```

class IqOffsetCls

IqOffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class IqOffsetStruct

Response structure. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- Offset_1: float: I/Q origin offset limit for high TX power range
- Offset_2: float: I/Q origin offset limit for intermediate TX power range
- Offset_3: float: I/Q origin offset limit for low TX power range

get() → IqOffsetStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IQOFfset
value: IqOffsetStruct = driver.configure.lteMeas.multiEval.limit.qpsk.iqOffset.
↪get()
```

Defines upper limits for the I/Q origin offset (QPSK modulation) . Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

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

set(*enable: bool*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:IQOFfset
driver.configure.lteMeas.multiEval.limit.qpsk.iqOffset.set(enable = False,␣
↪offset_1 = 1.0, offset_2 = 1.0, offset_3 = 1.0)
```

Defines upper limits for the I/Q origin offset (QPSK modulation) . Three different I/Q origin offset limits can be set for three TX power ranges. For details, see 'I/Q origin offset limits'.

param enable
OFF: disables the limit check ON: enables the limit check

param offset_1
I/Q origin offset limit for high TX power range

param offset_2
I/Q origin offset limit for intermediate TX power range

param offset_3
I/Q origin offset limit for low TX power range

6.1.1.4.3.33 Merror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
```

class MerrorCls

Merror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MerrorStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get() → MerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
value: MerrorStruct = driver.configure.lteMeas.multiEval.limit.qpsk.merror.get()
```

Defines upper limits for the RMS and peak values of the magnitude error for QPSK.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
driver.configure.lteMeas.multiEval.limit.qpsk.merror.set(rms = 1.0, peak = 1.0)
```

Defines upper limits for the RMS and peak values of the magnitude error for QPSK.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.4.3.34 Perror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
```

class PerrorCls

Perror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PerrorStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get() → PerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
value: PerrorStruct = driver.configure.lteMeas.multiEval.limit.qpsk.perror.get()
```

Defines symmetric limits for the RMS and peak values of the phase error for QPSK. The limit check fails if the absolute value of the measured phase error exceeds the specified limit.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
driver.configure.lteMeas.multiEval.limit.qpsk.perror.set(rms = 1.0, peak = 1.0)
```

Defines symmetric limits for the RMS and peak values of the phase error for QPSK. The limit check fails if the absolute value of the measured phase error exceeds the specified limit.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.4.3.35 SeMask

class SeMaskCls

SeMask commands group definition. 13 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.36 AtTolerance<EutraBand>

RepCap Settings

```
# Range: Nr30 .. Nr50
rc = driver.configure.lteMeas.multiEval.limit.seMask.atTolerance.repcap_eutraBand_get()
driver.configure.lteMeas.multiEval.limit.seMask.atTolerance.repcap_eutraBand_set(repcap.
↪EutraBand.Nr30)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:ATTolerance<EUTRAband>
```

class AtToleranceCls

AtTolerance commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: EutraBand, default value after init: EutraBand.Nr30

get(*eutraBand=EutraBand.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:ATTolerance
↪<EUTRAband>
value: float = driver.configure.lteMeas.multiEval.limit.seMask.atTolerance.
↪get(eutraBand = repcap.EutraBand.Default)
```

Defines additional test tolerances for the emission masks. The tolerance is added to the power values of all general and additional spectrum emission masks. A positive tolerance value relaxes the limits. For operating bands below 3 GHz, there is no additional test tolerance. You can define different additional test tolerances for bands above 3 GHz and for bands above 5 GHz.

param eutraBand
optional repeated capability selector. Default value: Nr30 (settable in the interface 'AtTolerance')

return
add_test_tol: Additional test tolerance

set(*add_test_tol: float*, *eutraBand=EutraBand.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:ATTolerance
↪<EUTRAband>
driver.configure.lteMeas.multiEval.limit.seMask.atTolerance.set(add_test_tol =␣
↪1.0, eutraBand = repcap.EutraBand.Default)
```

Defines additional test tolerances for the emission masks. The tolerance is added to the power values of all general and additional spectrum emission masks. A positive tolerance value relaxes the limits. For operating bands below 3 GHz, there is no additional test tolerance. You can define different additional test tolerances for bands above 3 GHz and for bands above 5 GHz.

param add_test_tol
Additional test tolerance

param eutraBand
optional repeated capability selector. Default value: Nr30 (settable in the interface 'AtTolerance')

Cloning the Group

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

6.1.1.4.3.37 Limit<Limit>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.repcap_limit_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.repcap_limit_set(repcap.Limit.Nr1)
```

class LimitCls

Limit commands group definition. 8 total commands, 3 Subgroups, 0 group commands Repeated Capability: Limit, default value after init: Limit.Nr1

Cloning the Group

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

Subgroups

6.1.1.4.3.38 Additional<Table>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.repcap_table_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.repcap_table_set(repcap.
↪Table.Nr1)
```

class AdditionalCls

Additional commands group definition. 4 total commands, 2 Subgroups, 0 group commands Repeated Capability: Table, default value after init: Table.Nr1

Cloning the Group

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

Subgroups

6.1.1.4.3.39 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.40 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.repcap_firstChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.repcap_firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.
↪carrierAggregation.channelBw1st.clone()
```

Subgroups

6.1.1.4.3.41 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.channelBw2nd.repcap_secondChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.carrierAggregation.
↪channelBw1st.channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Start frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Frequency_End: float: Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Level: float: Upper limit for the area
- Rbw: enums.Rbw: Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.lteMeas.multiEval.limit.seMask.
↪limit.additional.carrierAggregation.channelBw1st.channelBw2nd.get(limit =␣
↪repcap.Limit.Default, table = repcap.Table.Default, firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

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

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

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *table=Table.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.
↪carrierAggregation.channelBw1st.channelBw2nd.set(enable = False, frequency_
↪start = 1.0, frequency_end = 1.0, level = 1.0, rbw = enums.Rbw.K030, limit =␣
↪repcap.Limit.Default, table = repcap.Table.Default, firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param enable
OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
Start frequency of the area, relative to the edges of the aggregated channel bandwidth.

param frequency_end
Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.

param level
Upper limit for the area

param rbw
Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

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

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

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.
↪carrierAggregation.channelBw1st.channelBw2nd.clone()
```

6.1.1.4.3.42 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CAGGregation:OCOMbination
```

class OcombinationCls

Ocombination commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class OcombinationStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Start frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Frequency_End: float: Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Level: float: Upper limit for the area.
- Rbw: enums.Rbw: Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*) → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.lteMeas.multiEval.limit.seMask.
↪limit.additional.carrierAggregation.ocombination.get(limit = repcap.Limit.
↪Default, table = repcap.Table.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

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

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

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *table=Table.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CAGGregation:OCOMbination
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.
↪carrierAggregation.ocombination.set(enable = False, frequency_start = 1.0,␣
↪frequency_end = 1.0, level = 1.0, rbw = enums.Rbw.K030, limit = repcap.Limit.
↪Default, table = repcap.Table.Default)
```

Defines additional requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param enable
OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
Start frequency of the area, relative to the edges of the aggregated channel bandwidth.

param frequency_end
Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.

param level
Upper limit for the area.

param rbw
Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

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

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

6.1.1.4.3.43 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.channelBw.repcap_
↪channelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.channelBw.repcap_
↪channelBw_set(repcap.ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Lower border of the area, relative to the edges of the channel bandwidth.
- Frequency_End: float: Upper border of the area, relative to the edges of the channel bandwidth.
- Level: float: Upper limit for the area
- Rbw: enums.RbwExtended: Resolution bandwidth to be used for the area. Only a subset of the values is allowed, depending on Table and Band, see table below. K030: 30 kHz K050: 50 kHz K100: 100 kHz K150: 150 kHz K200: 200 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.lteMeas.multiEval.limit.seMask.limit.
↪additional.channelBw.get(limit = repcap.Limit.Default, table = repcap.Table.
↪Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for uplink measurements. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>. Several tables of additional requirements are available.

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

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

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: RbwExtended*, *limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.channelBw.
```

(continues on next page)

(continued from previous page)

```
↪set(enable = False, frequency_start = 1.0, frequency_end = 1.0, level = 1.0,␣
↪rbw = enums.RbwExtended.K030, limit = repcap.Limit.Default, table = repcap.
↪Table.Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for uplink measurements. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>. Several tables of additional requirements are available.

param enable
OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
Lower border of the area, relative to the edges of the channel bandwidth.

param frequency_end
Upper border of the area, relative to the edges of the channel bandwidth.

param level
Upper limit for the area

param rbw
Resolution bandwidth to be used for the area. Only a subset of the values is allowed, depending on Table and Band, see table below. K030: 30 kHz K050: 50 kHz K100: 100 kHz K150: 150 kHz K200: 200 kHz M1: 1 MHz

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

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

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

Subgroups

6.1.1.4.3.44 Sidelink

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:ADDitional<Table>
↪:CBANdwidth<Band>:SIDelink
```

class SidelinkCls

Sidelink commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SidelinkStruct

Structure for setting input parameters. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Lower border of the area, relative to the edges of the channel bandwidth.
- Frequency_End: float: Upper border of the area, relative to the edges of the channel bandwidth.
- Level: float: Upper limit at FrequencyStart
- Slope: float: Slope for the upper limit within the area
- Rbw: enums.Rbw: Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → SidelinkStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>:SIDelink
value: SidelinkStruct = driver.configure.lteMeas.multiEval.limit.seMask.limit.
↪additional.channelBw.sidelink.get(limit = repcap.Limit.Default, table =␣
↪repcap.Table.Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for sidelink measurements. The activation state, the area borders, the start value and slope of the upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

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

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

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*structure: SidelinkStruct*, *limit=Limit.Default*, *table=Table.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:ADDitional<Table>:CBANdwidth<Band>:SIDelink
structure = driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.
↪channelBw.sidelink.SidelinkStruct()
structure.Enable: bool = False
structure.Frequency_Start: float = 1.0
structure.Frequency_End: float = 1.0
structure.Level: float = 1.0
structure.Slope: float = 1.0
```

(continues on next page)

(continued from previous page)

```
structure.Rbw: enums.Rbw = enums.Rbw.K030
driver.configure.lteMeas.multiEval.limit.seMask.limit.additional.channelBw.
↪sidelink.set(structure, limit = repcap.Limit.Default, table = repcap.Table.
↪Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area <no>, for sidelink measurements. The activation state, the area borders, the start value and slope of the upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

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

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

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

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

6.1.1.4.3.45 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.46 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.repcap_firstChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.clone()
```

Subgroups

6.1.1.4.3.47 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.channelBw2nd.repcap_secondChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

class ChannelBw2ndStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Start frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Frequency_End: float: Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Level: float: Upper limit for the area
- Rbw: enums.Rbw: Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → ChannelBw2ndStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
value: ChannelBw2ndStruct = driver.configure.lteMeas.multiEval.limit.seMask.
↪limit.carrierAggregation.channelBw1st.channelBw2nd.get(limit = repcap.Limit.
↪Default, firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw =␣
↪repcap.SecondChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

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

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
structure: for return value, see the help for ChannelBw2ndStruct structure arguments.

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.channelBw2nd.set(enable = False, frequency_start = 1.0,␣
↪frequency_end = 1.0, level = 1.0, rbw = enums.Rbw.K030, limit = repcap.Limit.
↪Default, firstChannelBw = repcap.FirstChannelBw.Default, secondChannelBw =␣
↪repcap.SecondChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param enable
OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
Start frequency of the area, relative to the edges of the aggregated channel bandwidth.

param frequency_end
Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.

param level
Upper limit for the area

param rbw
Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

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

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.channelBw2nd.clone()
```

Subgroups

6.1.1.4.3.48 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

class ChannelBw3rdStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Start frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Frequency_End: float: Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Level: float: Upper limit for the area
- Rbw: enums.Rbw: Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → ChannelBw3rdStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
value: ChannelBw3rdStruct = driver.configure.lteMeas.multiEval.limit.seMask.
↪limit.carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(limit =␣
↪repcap.Limit.Default, firstChannelBw = repcap.FirstChannelBw.Default,␣
↪secondChannelBw = repcap.SecondChannelBw.Default, thirdChannelBw = repcap.
↪ThirdChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

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

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

return
structure: for return value, see the help for ChannelBw3rdStruct structure arguments.

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>:CBANdwidth<Band3>
driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.set(enable = False, frequency_start =␣
↪1.0, frequency_end = 1.0, level = 1.0, rbw = enums.Rbw.K030, limit = repcap.
↪Limit.Default, firstChannelBw = repcap.FirstChannelBw.Default,␣
↪secondChannelBw = repcap.SecondChannelBw.Default, thirdChannelBw = repcap.
↪ThirdChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param enable
OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
Start frequency of the area, relative to the edges of the aggregated channel bandwidth.

param frequency_end
Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.

param level
Upper limit for the area

param rbw
Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

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

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw3rd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.clone()
```

6.1.1.4.3.49 Ocombination

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:OCOMbination
```

class OcombinationCls

Ocombination commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class OcombinationStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Start frequency of the area, relative to the edges of the aggregated channel bandwidth.
- Frequency_End: float: Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.

- Level: float: Upper limit for the area
- Rbw: enums.Rbw: Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*) → OcombinationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:OCOMbination
value: OcombinationStruct = driver.configure.lteMeas.multiEval.limit.seMask.
↪limit.carrierAggregation.ocombination.get(limit = repcap.Limit.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

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

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CAGGregation:OCOMbination
driver.configure.lteMeas.multiEval.limit.seMask.limit.carrierAggregation.
↪ocombination.set(enable = False, frequency_start = 1.0, frequency_end = 1.0,␣
↪level = 1.0, rbw = enums.Rbw.K030, limit = repcap.Limit.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The settings apply to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param enable
OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
Start frequency of the area, relative to the edges of the aggregated channel bandwidth.

param frequency_end
Stop frequency of the area, relative to the edges of the aggregated channel bandwidth.

param level
Upper limit for the area

param rbw
Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

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

6.1.1.4.3.50 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.limit.channelBw.repcap_channelBw_
↪get()
driver.configure.lteMeas.multiEval.limit.seMask.limit.channelBw.repcap_channelBw_
↪set(repcap.ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Lower border of the area, relative to the edges of the channel bandwidth.
- Frequency_End: float: Upper border of the area, relative to the edges of the channel bandwidth.
- Level: float: Upper limit for the area
- Rbw: enums.Rbw: Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

get(*limit=Limit.Default*, *channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.lteMeas.multiEval.limit.seMask.limit.
↪channelBw.get(limit = repcap.Limit.Default, channelBw = repcap.ChannelBw.
↪Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

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

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: Rbw*, *limit=Limit.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:LIMit<nr>
↪:CBANdwidth<Band>
driver.configure.lteMeas.multiEval.limit.seMask.limit.channelBw.set(enable =␣
↪False, frequency_start = 1.0, frequency_end = 1.0, level = 1.0, rbw = enums.
↪Rbw.K030, limit = repcap.Limit.Default, channelBw = repcap.ChannelBw.Default)
```

Defines general requirements for the emission mask area <no>. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <Band>.

param enable
OFF: Disables the check of these requirements. ON: Enables the check of these requirements.

param frequency_start
Lower border of the area, relative to the edges of the channel bandwidth.

param frequency_end
Upper border of the area, relative to the edges of the channel bandwidth.

param level
Upper limit for the area

param rbw
Resolution bandwidth to be used for the area. K030: 30 kHz K100: 100 kHz M1: 1 MHz

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

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.1.4.3.51 ObwLimit

class ObwLimitCls

ObwLimit commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.3.52 CarrierAggregation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:OCOMbination
```

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_ocombination() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:OCOMbination
value: float or bool = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.
↪carrierAggregation.get_ocombination()
```

Defines an upper limit for the occupied bandwidth. The setting applies to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

return
obw_limit: (float or boolean) No help available

set_ocombination(*obw_limit: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:OCOMbination
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.set_
↪ocombination(obw_limit = 1.0)
```

Defines an upper limit for the occupied bandwidth. The setting applies to all 'other' channel bandwidth combinations, not covered by other commands in this chapter.

param obw_limit
(float or boolean) No help available

Cloning the Group

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

Subgroups

6.1.1.4.3.53 ChannelBw1st<FirstChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.repcap_firstChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪repcap_firstChannelBw_set(repcap.FirstChannelBw.Bw100)
```

class ChannelBw1stCls

ChannelBw1st commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: FirstChannelBw, default value after init: FirstChannelBw.Bw100

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.clone()
```

Subgroups

6.1.1.4.3.54 ChannelBw2nd<SecondChannelBw>

RepCap Settings

```
# Range: Bw50 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.repcap_secondChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪channelBw2nd.repcap_secondChannelBw_set(repcap.SecondChannelBw.Bw50)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
```

class ChannelBw2ndCls

ChannelBw2nd commands group definition. 2 total commands, 1 Subgroups, 1 group commands Repeated Capability: SecondChannelBw, default value after init: SecondChannelBw.Bw50

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>
```

(continues on next page)

(continued from previous page)

```
value: float or bool = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.
↪carrierAggregation.channelBw1st.channelBw2nd.get(firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The setting is defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

return
obw_limit: (float or boolean) No help available

set(*obw_limit: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.set(obw_limit = 1.0, firstChannelBw = repcap.
↪FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The setting is defined separately for each channel bandwidth combination, for two aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use ...:CBANdwidth100:CBANdwidth50.

param obw_limit
(float or boolean) No help available

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface 'ChannelBw1st')

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface 'ChannelBw2nd')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.clone()
```

Subgroups

6.1.1.4.3.55 ChannelBw3rd<ThirdChannelBw>

RepCap Settings

```
# Range: Bw100 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.repcap_thirdChannelBw_get()
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.channelBw1st.
↪channelBw2nd.channelBw3rd.repcap_thirdChannelBw_set(repcap.ThirdChannelBw.Bw100)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth<Band2>
↪:CBANdwidth<Band3>
```

class ChannelBw3rdCls

ChannelBw3rd commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ThirdChannelBw, default value after init: ThirdChannelBw.Bw100

get(*firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>:CBANdwidth<Band3>
value: float or bool = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.
↪carrierAggregation.channelBw1st.channelBw2nd.channelBw3rd.get(firstChannelBw␣
↪= repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface ‘ChannelBw1st’)

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface ‘ChannelBw2nd’)

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface ‘ChannelBw3rd’)

return
obw_limit: (float or boolean) No help available

set(*obw_limit: float*, *firstChannelBw=FirstChannelBw.Default*, *secondChannelBw=SecondChannelBw.Default*, *thirdChannelBw=ThirdChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CAGGregation:CBANdwidth<Band1>:CBANdwidth
↪<Band2>:CBANdwidth<Band3>
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.set(obw_limit = 1.0, firstChannelBw =␣
↪repcap.FirstChannelBw.Default, secondChannelBw = repcap.SecondChannelBw.
↪Default, thirdChannelBw = repcap.ThirdChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth. The settings are defined separately for each channel bandwidth combination, for three aggregated carriers. The following bandwidth combinations are supported: Example: For the first line in the figure, use …:CBANdwidth200:CBANdwidth150:CBANdwidth100.

param obw_limit
(float or boolean) No help available

param firstChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface ‘ChannelBw1st’)

param secondChannelBw
optional repeated capability selector. Default value: Bw50 (settable in the interface ‘ChannelBw2nd’)

param thirdChannelBw
optional repeated capability selector. Default value: Bw100 (settable in the interface ‘ChannelBw3rd’)

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.carrierAggregation.
↪channelBw1st.channelBw2nd.channelBw3rd.clone()
```

6.1.1.4.3.56 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.channelBw.repcap_channelBw_
↪get()
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.channelBw.repcap_channelBw_
↪set(repcap.ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

get(*channelBw=ChannelBw.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<Band>
value: float or bool = driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.
↪channelBw.get(channelBw = repcap.ChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth, depending on the channel bandwidth.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

return
obw_limit: (float or boolean) No help available

set(*obw_limit: float*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<Band>
driver.configure.lteMeas.multiEval.limit.seMask.obwLimit.channelBw.set(obw_
↪limit = 1.0, channelBw = repcap.ChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth, depending on the channel bandwidth.

param obw_limit
(float or boolean) No help available

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.1.4.4 ListPy

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:OSINdex
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:PLCMode
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMODe
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:NCONnections
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST
```

class ListPyCls

ListPy commands group definition. 26 total commands, 3 Subgroups, 5 group commands

get_cmode() → ParameterSetMode

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

Sets the connector mode, selecting whether all list mode segments use the same RF connection.

return

connector_mode: - GLOBal: Use the same RF connection for all segments, see ROUTe:LTE:MEASi:SPATh. - LIST: Assign a connection to each segment, see CONFigure:LTE:MEASi:MEValuation:LIST:SEGMentno:CIDX.

get_nconnections() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:NCONnections
value: int = driver.configure.lteMeas.multiEval.listPy.get_nconnections()
```

Sets the number of connections to be defined for the list mode, for connector mode LIST. Define the connections via ROUTe:LTE:MEAS<i>:SPATh.

return

no_of_connections: The maximum number of connections is limited by the number of connectors per smart channel.

get_os_index() → int

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

No command help available

return

offline_seg_index: (integer or boolean) No help available

get_plc_mode() → ParameterSetMode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:PLCMode
value: enums.ParameterSetMode = driver.configure.lteMeas.multiEval.listPy.get_
↪plc_mode()
```

Selects which physical cell ID setting is used for list mode measurements.

return
plc_id_mode: - GLOBal: The global setting is used for all segments, see CONFigure:LTE:MEASi:MEValuation:CCno:PLCid. - LIST: The cell ID is configured per segment, see CONFigure:LTE:MEASi:MEValuation:LIST:SEGMentno:PLCid.

get_value() → bool

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

Enables or disables the list mode.

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

set_cmode(*connector_mode: ParameterSetMode*) → None

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

Sets the connector mode, selecting whether all list mode segments use the same RF connection.

param connector_mode

- GLOBal: Use the same RF connection for all segments, see ROUTe:LTE:MEASi:SPATh.
- LIST: Assign a connection to each segment, see CONFigure:LTE:MEASi:MEValuation:LIST:SEGMentno:CIDX.

set_nconnections(*no_of_connections: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:NCONnections
driver.configure.lteMeas.multiEval.listPy.set_nconnections(no_of_connections =␣
↪1)
```

Sets the number of connections to be defined for the list mode, for connector mode LIST. Define the connections via ROUTe:LTE:MEAS<i>:SPATh.

param no_of_connections
The maximum number of connections is limited by the number of connectors per smart channel.

set_os_index(*offline_seg_index: int*) → None

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

No command help available

param offline_seg_index
(integer or boolean) No help available

set_plc_mode(*plc_id_mode: ParameterSetMode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:PLCMode
driver.configure.lteMeas.multiEval.listPy.set_plc_mode(plc_id_mode = enums.
↪ParameterSetMode.GLOBal)
```

Selects which physical cell ID setting is used for list mode measurements.

param plc_id_mode

- GLOBal: The global setting is used for all segments, see CONFigure:LTE:MEASi:MEValuation:CCno:PLCid.
- LIST: The cell ID is configured per segment, see CONFigure:LTE:MEASi:MEValuation:LIST:SEGMentno:PLCid.

set_value(*enable: bool*) → None

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

Enables or disables the list mode.

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

Cloning the Group

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

Subgroups

6.1.1.4.4.1 Lrange

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:LRANge
```

class LrangeCls

Lrange commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class LrangeStruct

Response structure. Fields:

- Start_Index: int: First measured segment in the range of configured segments
- Nr_Segments: int: Number of measured segments

get() → LrangeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:LRANge
value: LrangeStruct = driver.configure.lteMeas.multiEval.listPy.lrange.get()
```

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

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

set(*start_index: int*, *nr_segments: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:LRANge
driver.configure.lteMeas.multiEval.listPy.lrange.set(start_index = 1, nr_
↪segments = 1)
```

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

param start_index
First measured segment in the range of configured segments

param nr_segments
Number of measured segments

6.1.1.4.4.2 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2000
rc = driver.configure.lteMeas.multiEval.listPy.segment.repcap_segment_get()
driver.configure.lteMeas.multiEval.listPy.segment.repcap_segment_set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 18 total commands, 15 Subgroups, 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.lteMeas.multiEval.listPy.segment.clone()
```

Subgroups

6.1.1.4.4.3 Aclr

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR
```

class AclrCls

Aclr commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class AclrStruct

Response structure. Fields:

- Aclr_Statistics: int: Statistical length in slots
- Aclr_Enable: bool: Enable or disable the measurement of ACLR results. ON: ACLR results are measured according to the other enable flags in this command. ACLR results for which there is no explicit enable flag are also measured (e.g. power in the assigned E-UTRA channel) . OFF: No ACLR results at all are measured. The other enable flags in this command are ignored.

- Utra_1_Enable: bool: Enable or disable evaluation of first adjacent UTRA channels.
- Utra_2_Enable: bool: Enable or disable evaluation of second adjacent UTRA channels.
- Eutra_Enable: bool: Enable or disable evaluation of first adjacent E-UTRA channels.

get(*segment=Segment.Default*) → AclrStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR
value: AclrStruct = driver.configure.lteMeas.multiEval.listPy.segment.aclr.
↪get(segment = repcap.Segment.Default)
```

Defines settings for ACLR measurements in list mode for segment <no>.

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

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

set(*aclr_statistics: int*, *aclr_enable: bool*, *utra_1_enable: bool*, *utra_2_enable: bool*, *eutra_enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR
driver.configure.lteMeas.multiEval.listPy.segment.aclr.set(aclr_statistics = 1,␣
↪aclr_enable = False, utra_1_enable = False, utra_2_enable = False, eutra_
↪enable = False, segment = repcap.Segment.Default)
```

Defines settings for ACLR measurements in list mode for segment <no>.

param aclr_statistics
Statistical length in slots

param aclr_enable
Enable or disable the measurement of ACLR results. ON: ACLR results are measured according to the other enable flags in this command. ACLR results for which there is no explicit enable flag are also measured (e.g. power in the assigned E-UTRA channel) . OFF: No ACLR results at all are measured. The other enable flags in this command are ignored.

param utra_1_enable
Enable or disable evaluation of first adjacent UTRA channels.

param utra_2_enable
Enable or disable evaluation of second adjacent UTRA channels.

param eutra_enable
Enable or disable evaluation of first adjacent E-UTRA channels.

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

6.1.1.4.4.4 CarrierAggregation

class CarrierAggregationCls

CarrierAggregation commands group definition. 3 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.4.5 AcSpacing

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CAGGregation:ACSPacing
```

class AcSpacingCls

AcSpacing commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:ACSPacing
driver.configure.lteMeas.multiEval.listPy.segment.carrierAggregation.acSpacing.
↪set(segment = repcap.Segment.Default)
```

Adjusts the component carrier frequencies in segment <no>, so that the carriers are aggregated contiguously.

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

set_with_opc(*segment=Segment.Default*, *opc_timeout_ms: int = -1*) → None

6.1.1.4.4.6 Mcarrier

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CAGGregation:MCARrier
```

class McarrierCls

Mcarrier commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get(*segment=Segment.Default*) → MeasCarrier

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier
value: enums.MeasCarrier = driver.configure.lteMeas.multiEval.listPy.segment.
↪carrierAggregation.mcarrier.get(segment = repcap.Segment.Default)
```

No command help available

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

return
meas_carrier: No help available

set(*meas_carrier: MeasCarrier*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier
driver.configure.lteMeas.multiEval.listPy.segment.carrierAggregation.mcarrier.
↪set(meas_carrier = enums.MeasCarrier.PCC, segment = repcap.Segment.Default)
```

No command help available

param meas_carrier
No help available

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

Cloning the Group

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

Subgroups

6.1.1.4.4.7 Enhanced

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier:ENHanced
```

class EnhancedCls

Enhanced commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segment=Segment.Default*) → MeasCarrierEnhanced

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier:ENHanced
value: enums.MeasCarrierEnhanced = driver.configure.lteMeas.multiEval.listPy.
↪segment.carrierAggregation.mcarrier.enhanced.get(segment = repcap.Segment.
↪Default)
```

Selects a component carrier for single carrier measurements in segment <no>.

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

return
meas_carrier: No help available

set(*meas_carrier: MeasCarrierEnhanced*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:CAGGregation:MCARrier:ENHanced
driver.configure.lteMeas.multiEval.listPy.segment.carrierAggregation.mcarrier.
↪enhanced.set(meas_carrier = enums.MeasCarrierEnhanced.CC1, segment = repcap.
↪Segment.Default)
```

Selects a component carrier for single carrier measurements in segment <no>.

param meas_carrier
No help available

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

6.1.1.4.4.8 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.lteMeas.multiEval.listPy.segment.cc.repcap_carrierComponent_get()
driver.configure.lteMeas.multiEval.listPy.segment.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CC<c>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

class CcStruct

Response structure. Fields:

- Frequency: float: Center frequency of the component carrier For the supported range, see 'Frequency ranges'.

- Ch_Bandwidth: enums.ChannelBandwidth: Channel bandwidth of the component carrier B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

get(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → CcStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CC<c>
value: CcStruct = driver.configure.lteMeas.multiEval.listPy.segment.cc.
↪get(segment = repcap.Segment.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Defines carrier-specific analyzer settings for component carrier CC<c>, in segment <no>. This command is only relevant for carrier aggregation.

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

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

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

set(*frequency: float*, *ch_bandwidth: ChannelBandwidth*, *segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CC<c>
driver.configure.lteMeas.multiEval.listPy.segment.cc.set(frequency = 1.0, ch_
↪bandwidth = enums.ChannelBandwidth.B014, segment = repcap.Segment.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Defines carrier-specific analyzer settings for component carrier CC<c>, in segment <no>. This command is only relevant for carrier aggregation.

param frequency
Center frequency of the component carrier For the supported range, see 'Frequency ranges'.

param ch_bandwidth
Channel bandwidth of the component carrier B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz

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

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

Cloning the Group

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

6.1.1.4.4.9 Cidx

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CIDX
```

class CidxCls

Cidx commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segment=Segment.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CIDX
value: int = driver.configure.lteMeas.multiEval.listPy.segment.cidx.get(segment␣
↪= repcap.Segment.Default)
```

Selects the RF connection index for segment <no>. To define the connection indices, see ROUTe:LTE:MEAS<i>:SPATh.

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

return
connection_index: No help available

set(*connection_index: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CIDX
driver.configure.lteMeas.multiEval.listPy.segment.cidx.set(connection_index = 1,
↪ segment = repcap.Segment.Default)
```

Selects the RF connection index for segment <no>. To define the connection indices, see ROUTe:LTE:MEAS<i>:SPATh.

param connection_index
No help available

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

6.1.1.4.4.10 Emtc

class EmtcCls

Emtc commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.4.11 Nband

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:EMTC:NBANd
```

class NbandCls

Nband commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segment=Segment.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:EMTC:NBANd
value: int = driver.configure.lteMeas.multiEval.listPy.segment.emtc.nband.
↪get(segment = repcap.Segment.Default)
```

Selects the eMTC narrowband for segment <no>.

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

return
number: The maximum depends on the channel BW, see 'RB allocation, narrowbands and widebands for eMTC'.

set(*number: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:EMTC:NBANd
driver.configure.lteMeas.multiEval.listPy.segment.emtc.nband.set(number = 1,␣
↪segment = repcap.Segment.Default)
```

Selects the eMTC narrowband for segment <no>.

param number
The maximum depends on the channel BW, see 'RB allocation, narrowbands and widebands for eMTC'.

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

6.1.1.4.4.12 Modulation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation
```

class ModulationCls

Modulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class ModulationStruct

Structure for setting input parameters. Fields:

- Mod_Statistics: int: Statistical length in slots.
- Modenable: bool: Enable or disable the measurement of modulation results. ON: Modulation results are measured according to the other enable flags in this command. Modulation results for which there is no explicit enable flag are also measured (e.g. I/Q offset, frequency error and timing error) . OFF: No modulation results at all are measured. The other enable flags in this command are ignored.
- Evm_Enable: bool: Enable or disable measurement of EVM.
- Mag_Error_Enable: bool: Enable or disable measurement of magnitude error.
- Phase_Err_Enable: bool: Enable or disable measurement of phase error.
- Ib_Eenable: bool: Enable or disable measurement of in-band emissions.
- Eq_Sp_Flat_Enable: bool: Enable or disable measurement of equalizer spectrum flatness results.
- Mod_Scheme: enums.ModScheme: The modulation scheme used by the LTE uplink signal. AUTO: automatic detection QPSK: QPSK Q16: 16QAM Q64: 64QAM Q256: 256QAM

get(*segment=Segment.Default*) → ModulationStruct

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

Defines settings for modulation measurements in list mode for segment <no>.

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: ModulationStruct*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation
structure = driver.configure.lteMeas.multiEval.listPy.segment.modulation.
↪ModulationStruct()
structure.Mod_Statistics: int = 1
structure.Modenable: bool = False
structure.Evm_Enable: bool = False
structure.Mag_Error_Enable: bool = False
```

(continues on next page)

(continued from previous page)

```
structure.Phase_Err_Enable: bool = False
structure.Ib_Eenable: bool = False
structure.Eq_Sp_Flat_Enable: bool = False
structure.Mod_Scheme: enums.ModScheme = enums.ModScheme.AUTO
driver.configure.lteMeas.multiEval.listPy.segment.modulation.set(structure,␣
↪segment = repcap.Segment.Default)
```

Defines settings for modulation measurements in list mode for segment <no>.

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')

6.1.1.4.4.13 PlcId

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PLCid
```

class PlcIdCls

PlcId commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segment=Segment.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PLCid
value: int = driver.configure.lteMeas.multiEval.listPy.segment.plcId.
↪get(segment = repcap.Segment.Default)
```

Specifies the physical cell ID for segment <no>. See also method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy. plcMode.

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

return
phys_layer_cell_id: No help available

set(*phys_layer_cell_id: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PLCid
driver.configure.lteMeas.multiEval.listPy.segment.plcId.set(phys_layer_cell_id␣
↪= 1, segment = repcap.Segment.Default)
```

Specifies the physical cell ID for segment <no>. See also method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy. plcMode.

param phys_layer_cell_id
No help available

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

6.1.1.4.4.14 Pmonitor

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor
```

class PmonitorCls

Pmonitor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segment=Segment.Default*) → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor
value: bool = driver.configure.lteMeas.multiEval.listPy.segment.pmonitor.
↪get(segment = repcap.Segment.Default)
```

Enables or disables the measurement of power monitor results (power of one carrier) for segment <no>.

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

return
enable: No help available

set(*enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor
driver.configure.lteMeas.multiEval.listPy.segment.pmonitor.set(enable = False,␣
↪segment = repcap.Segment.Default)
```

Enables or disables the measurement of power monitor results (power of one carrier) for segment <no>.

param enable
No help available

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

6.1.1.4.4.15 Power

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer
```

class PowerCls

Power commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PowerStruct

Response structure. Fields:

- Power_Statistics: int: Statistical length in subframes
- Power_Enable: bool: Enables or disables the measurement of the total TX power.

get(*segment=Segment.Default*) → PowerStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer
value: PowerStruct = driver.configure.lteMeas.multiEval.listPy.segment.power.
↪get(segment = repcap.Segment.Default)
```

Defines settings for the measurement of the total TX power of all carriers for segment <no>.

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

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

set(*power_statistics: int*, *power_enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer
driver.configure.lteMeas.multiEval.listPy.segment.power.set(power_statistics =␣
↪1, power_enable = False, segment = repcap.Segment.Default)
```

Defines settings for the measurement of the total TX power of all carriers for segment <no>.

param power_statistics
Statistical length in subframes

param power_enable
Enables or disables the measurement of the total TX power.

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

6.1.1.4.4.16 RbAllocation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:RBALlocation
```

class RbAllocationCls

RbAllocation commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class RbAllocationStruct

Response structure. Fields:

- Auto: bool: OFF: manual definition via NoRB and Offset ON: automatic detection of RB allocation
- No_Rb: int: Number of allocated resource blocks in each measured slot
- Offset: int: Offset of first allocated resource block from edge of allocated UL transmission bandwidth

get(*segment=Segment.Default*) → RbAllocationStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation
value: RbAllocationStruct = driver.configure.lteMeas.multiEval.listPy.segment.
↪rbAllocation.get(segment = repcap.Segment.Default)
```

Defines the uplink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically.

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

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

set(*auto: bool*, *no_rb: int*, *offset: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation
driver.configure.lteMeas.multiEval.listPy.segment.rbAllocation.set(auto = False,
↪ no_rb = 1, offset = 1, segment = repcap.Segment.Default)
```

Defines the uplink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically.

param auto
OFF: manual definition via NoRB and Offset ON: automatic detection of RB allocation

param no_rb
Number of allocated resource blocks in each measured slot

param offset
Offset of first allocated resource block from edge of allocated UL transmission bandwidth

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

Cloning the Group

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

Subgroups

6.1.1.4.4.17 Sidelink

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:RBALlocation:SIDelink
```

class SidelinkCls

Sidelink commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SidelinkStruct

Response structure. Fields:

- Auto: bool: OFF: manual definition via the other settings ON: automatic detection of RB allocation
- No_Rb_Pssch: int: Number of allocated RBs for the PSSCH in each measured slot
- Offset_Pssch: int: Offset of the first allocated PSSCH resource block
- Offset_Pscch: int: Offset of the first allocated PSCCH resource block

get(*segment=Segment.Default*) → SidelinkStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation:SIDelink
value: SidelinkStruct = driver.configure.lteMeas.multiEval.listPy.segment.
↪rbAllocation.sidelink.get(segment = repcap.Segment.Default)
```

Defines the sidelink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically. Most allowed input ranges depend on other settings, see 'Sidelink resource block allocation'.

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

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

set(*auto: bool*, *no_rb_pssch: int*, *offset_pssch: int*, *offset_pscch: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:RBALlocation:SIDelink
driver.configure.lteMeas.multiEval.listPy.segment.rbAllocation.sidelink.
↪set(auto = False, no_rb_pssch = 1, offset_pssch = 1, offset_pscch = 1,␣
↪segment = repcap.Segment.Default)
```

Defines the sidelink resource block allocation manually for segment <no>. By default, the RB allocation is detected automatically. Most allowed input ranges depend on other settings, see 'Sidelink resource block allocation'.

param auto
OFF: manual definition via the other settings ON: automatic detection of RB allocation

param no_rb_pssch
Number of allocated RBs for the PSSCH in each measured slot

param offset_pssch
Offset of the first allocated PSSCH resource block

param offset_pscch
Offset of the first allocated PSCCH resource block

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

6.1.1.4.4.18 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.lteMeas.multiEval.listPy.segment.scc.repcap_secondaryCC_get()
driver.configure.lteMeas.multiEval.listPy.segment.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SCC<c>
```

class SccCls

Scc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

class SccStruct

Response structure. Fields:

- Frequency: float: No parameter help available
- Ch_Bandwidth: enums.ChannelBandwidth: No parameter help available

get(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → SccStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SCC<c>
value: SccStruct = driver.configure.lteMeas.multiEval.listPy.segment.scc.
↪get(segment = repcap.Segment.Default, secondaryCC = repcap.SecondaryCC.
↪Default)
```

No command help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

set(*frequency: float*, *ch_bandwidth: ChannelBandwidth*, *segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SCC<c>
driver.configure.lteMeas.multiEval.listPy.segment.scc.set(frequency = 1.0, ch_
↪bandwidth = enums.ChannelBandwidth.B014, segment = repcap.Segment.Default,␣
↪secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param frequency
No help available

param ch_bandwidth
No help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

Cloning the Group

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

6.1.1.4.4.19 SeMask

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask
```

class SeMaskCls

SeMask commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SeMaskStruct

Response structure. Fields:

- Sem_Statistics: int: Statistical length in slots.
- Se_Enable: bool: Enable or disable the measurement of spectrum emission results. ON: Spectrum emission results are measured according to the other ...enable flags in this command. Results for which there is no explicit enable flag are also measured. OFF: No spectrum emission results at all are measured. The other enable flags in this command are ignored.
- Obw_Enable: bool: Enable or disable measurement of occupied bandwidth.
- Sem_Enable: bool: Enable or disable measurement of spectrum emission trace and margin results.

get(*segment=Segment.Default*) → SeMaskStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask
value: SeMaskStruct = driver.configure.lteMeas.multiEval.listPy.segment.seMask.
↪get(segment = repcap.Segment.Default)
```

Defines settings for spectrum emission measurements in list mode for segment <no>.

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

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

set(*sem_statistics: int*, *se_enable: bool*, *obw_enable: bool*, *sem_enable: bool*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask
driver.configure.lteMeas.multiEval.listPy.segment.seMask.set(sem_statistics = 1,
↪ se_enable = False, obw_enable = False, sem_enable = False, segment = repcap.
↪Segment.Default)
```

Defines settings for spectrum emission measurements in list mode for segment <no>.

param sem_statistics
Statistical length in slots.

param se_enable
Enable or disable the measurement of spectrum emission results. ON: Spectrum emission results are measured according to the other ...enable flags in this command. Results for which there is no explicit enable flag are also measured. OFF: No spectrum emission results at all are measured. The other enable flags in this command are ignored.

param obw_enable
Enable or disable measurement of occupied bandwidth.

param sem_enable
Enable or disable measurement of spectrum emission trace and margin results.

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

6.1.1.4.4.20 Setup

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SETup
```

class SetupCls

Setup commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class SetupStruct

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

- Segment_Length: int: Number of subframes in the segment
- Level: float: Expected nominal power in the segment. The range can be calculated as follows: Range (Expected Nominal Power) = Range (Input Power) + External Attenuation - User Margin The input power range is stated in the specifications document.
- Duplex_Mode: enums.Mode: Duplex mode used in the segment
- Band: enums.Band: TDD UL: OB33 | ... | OB45 | OB48 | OB50 | ... | OB53 | OB250 Sidelink: OB47 Operating band used in the segment
- Frequency: float: Center frequency of CC1 used in the segment For the supported range, see 'Frequency ranges'.
- Ch_Bandwidth: enums.ChannelBandwidth: Channel bandwidth of CC1 used in the segment. B014: 1.4 MHz B030: 3 MHz B050: 5 MHz B100: 10 MHz B150: 15 MHz B200: 20 MHz
- Cyclic_Prefix: enums.CyclicPrefix: Type of cyclic prefix used in the segment
- Channel_Type: enums.SegmentChannelTypeExtended: Channel type to be measured in the segment (AUTO for automatic detection) . Uplink: AUTO, PUSCh, PUCCh Sidelink: PSSCh, PSCCh, PSBCh
- **Retrigger_Flag: enums.RetriggerFlag: Specifies whether the measurement waits for a trigger event before measuring the segment, or not. The retrigger flag is ignored for trigger mode ONCE and evaluated for trigger mode SEGMent, see [CMDLINKRESOLVED Trigger.LteMeas.MultiEval.ListPy#Mode CMDLINKRESOLVED].**
  - OFF: Measure the segment without retrigger. For the first segment, the value OFF is interpreted as ON.
  - ON: Wait for a trigger event from the trigger source configured via TRIGger:LTE:MEASi:MEValuation:SOURce.
  - IFPower: Wait for a trigger event from the trigger source IF Power.The trigger evaluation bandwidth is 160 MHz.
  - IFPNarrowband: Wait for a trigger event from the trigger source IF Power.The trigger evaluation bandwidth is configured via TRIGger:LTE:MEASi:MEValuation:LIST:NBANdwidth.
- Evaluat_Offset: int: Number of subframes at the beginning of the segment that are not evaluated
- Network_Sig_Value: enums.NetworkSigValueNoCarrAggr: Optional setting parameter. Network signaled value to be used

for the segment

get(*segment*=*Segment.Default*) → SetupStruct

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

Defines the length and analyzer settings of segment <no>. This command must be sent for all segments to be measured (method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Lrange.set) . For uplink signals with TDD mode, see also method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Segment.Tdd.set. For carrier-specific settings for carrier aggregation, see CONFigure:LTE:MEAS<i>:MEValuation:LIST:SEGMent<no>:CC<c>.

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: SetupStruct*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SETup
structure = driver.configure.lteMeas.multiEval.listPy.segment.setup.
↪SetupStruct()
structure.Segment_Length: int = 1
structure.Level: float = 1.0
structure.Duplex_Mode: enums.Mode = enums.Mode.FDD
structure.Band: enums.Band = enums.Band.OB1
structure.Frequency: float = 1.0
structure.Ch_Bandwidth: enums.ChannelBandwidth = enums.ChannelBandwidth.B014
structure.Cyclic_Prefix: enums.CyclicPrefix = enums.CyclicPrefix.EXTended
structure.Channel_Type: enums.SegmentChannelTypeExtended = enums.
↪SegmentChannelTypeExtended.AUTO
structure.Retrigger_Flag: enums.RetriggerFlag = enums.RetriggerFlag.IFPNarrow
structure.Evaluat_Offset: int = 1
structure.Network_Sig_Value: enums.NetworkSigValueNoCarrAggr = enums.
↪NetworkSigValueNoCarrAggr.NS01
driver.configure.lteMeas.multiEval.listPy.segment.setup.set(structure, segment␣
↪= repcap.Segment.Default)
```

Defines the length and analyzer settings of segment <no>. This command must be sent for all segments to be measured (method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Lrange.set) . For uplink signals with TDD mode, see also method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Segment.Tdd.set. For carrier-specific settings for carrier aggregation, see CONFigure:LTE:MEAS<i>:MEValuation:LIST:SEGMent<no>:CC<c>.

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')

6.1.1.4.4.21 SingleCmw

class SingleCmwCls

SingleCmw commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.4.22 Connector

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:CMWS:CONNector
```

class ConnectorCls

Connector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*segment=Segment.Default*) → CmwsConnector

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

No command help available

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

return
cmws_connector: No help available

set(*cmws_connector: CmwsConnector*, *segment=Segment.Default*) → None

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

No command help available

param cmws_connector
No help available

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

6.1.1.4.4.23 Tdd

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:TDD
```

class TddCls

Tdd commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class TddStruct

Response structure. Fields:

- Uplink_Downlink: int: UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame
- Special_Subframe: int: Special subframe configuration, defining the inner structure of special subframes

get(*segment=Segment.Default*) → TddStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:TDD
value: TddStruct = driver.configure.lteMeas.multiEval.listPy.segment.tdd.
↪get(segment = repcap.Segment.Default)
```

Defines segment settings only relevant for uplink measurements with the duplex mode TDD. For general segment configuration, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Segment.Setup.set.

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

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

set(*uplink_downlink: int*, *special_subframe: int*, *segment=Segment.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:TDD
driver.configure.lteMeas.multiEval.listPy.segment.tdd.set(uplink_downlink = 1,␣
↪special_subframe = 1, segment = repcap.Segment.Default)
```

Defines segment settings only relevant for uplink measurements with the duplex mode TDD. For general segment configuration, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Segment.Setup.set.

param uplink_downlink
UL-DL configuration, defining the combination of uplink, downlink and special subframes within a radio frame

param special_subframe
Special subframe configuration, defining the inner structure of special subframes

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

6.1.1.4.4.24 SingleCmw

SCPI Command :

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

class SingleCmwCls

SingleCmw commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_cmode() → ParameterSetMode

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

No command help available

return
connector_mode: No help available

set_cmode(*connector_mode: ParameterSetMode*) → None

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

No command help available

param connector_mode
No help available

Cloning the Group

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

Subgroups

6.1.1.4.4.25 Connector

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMWS:CONNector:ALL
```

class ConnectorCls

Connector commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_all() → List[CmwsConnector]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMWS:CONNector:ALL
value: List[enums.CmwsConnector] = driver.configure.lteMeas.multiEval.listPy.
↪singleCmw.connector.get_all()
```

No command help available

return
cmws_connector: No help available

set_all(*cmws_connector: List[CmwsConnector]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:LIST:CMWS:CONNector:ALL
driver.configure.lteMeas.multiEval.listPy.singleCmw.connector.set_all(cmws_
↪connector = [CmwsConnector.R11, CmwsConnector.RB8])
```

No command help available

param cmws_connector
No help available

6.1.1.4.5 Modulation

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EQUalizer
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:MSCHeme
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:LLOCation
```

class ModulationCls

Modulation commands group definition. 9 total commands, 3 Subgroups, 3 group commands

get_equalizer() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EQUalizer
value: bool = driver.configure.lteMeas.multiEval.modulation.get_equalizer()
```

Enables or disables the post-FFT equalization step for the measurement of modulation results.

return
enable: No help available

get_llocation() → LocalOscLocation

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:LLOCation
value: enums.LocalOscLocation = driver.configure.lteMeas.multiEval.modulation.
↪get_llocation()
```

Specifies the UE transmitter architecture (local oscillator location) used for eMTC.

return
value: CN: Center of narrowband/wideband CCB: Center of channel bandwidth

get_mscheme() → ModScheme

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:MSCHeme
value: enums.ModScheme = driver.configure.lteMeas.multiEval.modulation.get_
↪mscheme()
```

Selects the modulation scheme used by the measured signal.

return
mod_scheme: Auto-detection, QPSK, 16QAM, 64QAM, 256QAM

set_equalizer(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EQUalizer
driver.configure.lteMeas.multiEval.modulation.set_equalizer(enable = False)
```

Enables or disables the post-FFT equalization step for the measurement of modulation results.

param enable
No help available

set_llocation(*value: LocalOscLocation*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:LLOCation
driver.configure.lteMeas.multiEval.modulation.set_llocation(value = enums.
↪LocalOscLocation.CCB)
```

Specifies the UE transmitter architecture (local oscillator location) used for eMTC.

param value
CN: Center of narrowband/wideband CCB: Center of channel bandwidth

set_mscheme(*mod_scheme: ModScheme*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:MSCHeme
driver.configure.lteMeas.multiEval.modulation.set_mscheme(mod_scheme = enums.
↪ModScheme.AUTO)
```

Selects the modulation scheme used by the measured signal.

param mod_scheme
Auto-detection, QPSK, 16QAM, 64QAM, 256QAM

Cloning the Group

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

Subgroups

6.1.1.4.5.1 CarrierAggregation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:CAGGregation:LLOCation
```

class CarrierAggregationCls

CarrierAggregation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_llocation() → CarrAggrLocalOscLocation

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:CAGGregation:LLOCation
value: enums.CarrAggrLocalOscLocation = driver.configure.lteMeas.multiEval.
↪modulation.carrierAggregation.get_llocation()
```

Specifies the UE transmitter architecture (local oscillator location) used for contiguous carrier aggregation.

return
value: CACB: Center of aggregated channel bandwidth CECC: Center of each component carrier

set_llocation(*value: CarrAggrLocalOscLocation*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:CAGGregation:LLOCation
driver.configure.lteMeas.multiEval.modulation.carrierAggregation.set_
↪llocation(value = enums.CarrAggrLocalOscLocation.AUTO)
```

Specifies the UE transmitter architecture (local oscillator location) used for contiguous carrier aggregation.

param value
CACB: Center of aggregated channel bandwidth CECC: Center of each component carrier

6.1.1.4.5.2 EePeriods

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUCCh
```

class EePeriodsCls

EePeriods commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_pucch() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUCCh
value: bool = driver.configure.lteMeas.multiEval.modulation.eePeriods.get_
↪pucch()
```

Enables or disables EVM exclusion periods for slots with detected channel type 'PUCCH'. If enabled, the first and the last SC-FDMA symbol of each slot is excluded from the calculation of EVM, magnitude error and phase error single value results. If the last symbol of a slot is already excluded because SRS signals are allowed, the second but last symbol is also excluded.

return
pucch: No help available

set_pucch(*pucch: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUCCh
driver.configure.lteMeas.multiEval.modulation.eePeriods.set_pucch(pucch = False)
```

Enables or disables EVM exclusion periods for slots with detected channel type 'PUCCH'. If enabled, the first and the last SC-FDMA symbol of each slot is excluded from the calculation of EVM, magnitude error and phase error single value results. If the last symbol of a slot is already excluded because SRS signals are allowed, the second but last symbol is also excluded.

param pucch
No help available

Cloning the Group

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

Subgroups

6.1.1.4.5.3 Pusch

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
```

class PuschCls

Pusch commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_lagging() → LaggingExclPeriod

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
value: enums.LaggingExclPeriod = driver.configure.lteMeas.multiEval.modulation.
↪eePeriods.pusch.get_lagging()
```

Specifies an EVM exclusion period at the end of each subframe (detected channel type 'PUSCH') ; if SRS signals are allowed, at the end of each shortened subframe. The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

return
lagging: OFF: no exclusion MS05: 5 s excluded MS25: 25 s excluded

get_leading() → LeadingExclPeriod

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
value: enums.LeadingExclPeriod = driver.configure.lteMeas.multiEval.modulation.
↪eePeriods.pusch.get_leading()
```

Specifies an EVM exclusion period at the beginning of a subframe (detected channel type 'PUSCH') . The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

return
leading: OFF: no exclusion MS25: 25 s excluded

set_lagging(*lagging: LaggingExclPeriod*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
driver.configure.lteMeas.multiEval.modulation.eePeriods.pusch.set_
↪lagging(lagging = enums.LaggingExclPeriod.MS05)
```

Specifies an EVM exclusion period at the end of each subframe (detected channel type 'PUSCH') ; if SRS signals are allowed, at the end of each shortened subframe. The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

param lagging
OFF: no exclusion MS05: 5 s excluded MS25: 25 s excluded

`set_leading`(*leading: LeadingExclPeriod*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
driver.configure.lteMeas.multiEval.modulation.eePeriods.pusch.set_
↪leading(leading = enums.LeadingExclPeriod.MS25)
```

Specifies an EVM exclusion period at the beginning of a subframe (detected channel type 'PUSCH') . The specified period is excluded from the calculation of EVM, magnitude error and phase error results.

param leading
OFF: no exclusion MS25: 25 s excluded

6.1.1.4.5.4 EwLength

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength
```

`class EwLengthCls`

EwLength commands group definition. 2 total commands, 1 Subgroups, 1 group commands

`class EwLengthStruct`

Response structure. Fields:

- Length_Cp_Normal: List[int]: No parameter help available
- Length_Cp_Extended: List[int]: No parameter help available

`get`() → EwLengthStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength
value: EwLengthStruct = driver.configure.lteMeas.multiEval.modulation.ewLength.
↪get()
```

Specifies the EVM window length in samples for all channel bandwidths, depending on the cyclic prefix (CP) type.

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

`set`(*length_cp_normal: List[int]*, *length_cp_extended: List[int]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength
driver.configure.lteMeas.multiEval.modulation.ewLength.set(length_cp_normal =␣
↪[1, 2, 3], length_cp_extended = [1, 2, 3])
```

Specifies the EVM window length in samples for all channel bandwidths, depending on the cyclic prefix (CP) type.

param length_cp_normal
No help available

param length_cp_extended
No help available

Cloning the Group

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

Subgroups

6.1.1.4.5.5 ChannelBw<ChannelBw>

RepCap Settings

```
# Range: Bw14 .. Bw200
rc = driver.configure.lteMeas.multiEval.modulation.ewLength.channelBw.repcap_channelBw_
↪get()
driver.configure.lteMeas.multiEval.modulation.ewLength.channelBw.repcap_channelBw_
↪set(repcap.ChannelBw.Bw14)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation:EWLength:CBANdwidth<Band>
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw14

class ChannelBwStruct

Response structure. Fields:

- Cyc_Prefix_Normal: int: Samples for normal CP
- Cyc_Prefix_Extend: int: Samples for extended CP

get(*channelBw=ChannelBw.Default*) → ChannelBwStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EWLength:CBANdwidth<Band>
value: ChannelBwStruct = driver.configure.lteMeas.multiEval.modulation.ewLength.
↪channelBw.get(channelBw = repcap.ChannelBw.Default)
```

Specifies the EVM window length in samples for a selected channel bandwidth, depending on the cyclic prefix (CP) type.

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

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

set(*cyc_prefix_normal: int*, *cyc_prefix_extend: int*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:MODulation:EWLength:CBANdwidth<Band>
driver.configure.lteMeas.multiEval.modulation.ewLength.channelBw.set(cyc_prefix_
↪normal = 1, cyc_prefix_extend = 1, channelBw = repcap.ChannelBw.Default)
```

Specifies the EVM window length in samples for a selected channel bandwidth, depending on the cyclic prefix (CP) type.

param cyc_prefix_normal
Samples for normal CP

param cyc_prefix_extend
Samples for extended CP

param channelBw
optional repeated capability selector. Default value: Bw14 (settable in the interface 'ChannelBw')

Cloning the Group

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

6.1.1.4.6 MsubFrames

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:MSUBframes
```

class MsubFramesCls

MsubFrames commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MsubFramesStruct

Response structure. Fields:

- Sub_Frame_Offset: int: Start of the measured subframe range relative to the trigger event.
- Sub_Frame_Count: int: Length of the measured subframe range.
- Meas_Subframe: int: Subframe containing the measured slots for modulation and spectrum results.

get() → MsubFramesStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSUBframes
value: MsubFramesStruct = driver.configure.lteMeas.multiEval.msubFrames.get()
```

Configures the scope of the measurement, i.e. which subframes are measured.

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

set(*sub_frame_offset: int*, *sub_frame_count: int*, *meas_subframe: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:MSUBframes
driver.configure.lteMeas.multiEval.msubFrames.set(sub_frame_offset = 1, sub_
↪frame_count = 1, meas_subframe = 1)
```

Configures the scope of the measurement, i.e. which subframes are measured.

param sub_frame_offset
Start of the measured subframe range relative to the trigger event.

param sub_frame_count
Length of the measured subframe range.

param meas_subframe
Subframe containing the measured slots for modulation and spectrum results.

6.1.1.4.7 NsValue

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue:CAGGregation
CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue
```

class NsValueCls

NsValue commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_carrier_aggregation() → NetworkSigValue

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue:CAGGregation
value: enums.NetworkSigValue = driver.configure.lteMeas.multiEval.nsValue.get_
↪carrier_aggregation()
```

Selects the 'network signaled value' for measurements with carrier aggregation.

return
value: Value CA_NS_01 to CA_NS_32

get_value() → NetworkSigValueNoCarrAggr

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue
value: enums.NetworkSigValueNoCarrAggr = driver.configure.lteMeas.multiEval.
↪nsValue.get_value()
```

Selects the 'network signaled value' for measurements without carrier aggregation. For Signal Path = Network, the setting is not configurable.

return
value: Value NS_01 to NS_288

set_carrier_aggregation(*value: NetworkSigValue*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue:CAGGregation
driver.configure.lteMeas.multiEval.nsValue.set_carrier_aggregation(value =␣
↪enums.NetworkSigValue.NS01)
```

Selects the 'network signaled value' for measurements with carrier aggregation.

param value
Value CA_NS_01 to CA_NS_32

set_value(*value: NetworkSigValueNoCarrAggr*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue
driver.configure.lteMeas.multiEval.nsValue.set_value(value = enums.
↪NetworkSigValueNoCarrAggr.NS01)
```

Selects the 'network signaled value' for measurements without carrier aggregation. For Signal Path = Network, the setting is not configurable.

param value
Value NS_01 to NS_288

6.1.1.4.8 Pcc

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation[:PCC]:PLCid
```

class PccCls

Pcc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_plc_id() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation[:PCC]:PLCid
value: int = driver.configure.lteMeas.multiEval.pcc.get_plc_id()
```

No command help available

return
phs_layer_cell_id: No help available

set_plc_id(*phs_layer_cell_id: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation[:PCC]:PLCid
driver.configure.lteMeas.multiEval.pcc.set_plc_id(phs_layer_cell_id = 1)
```

No command help available

param phs_layer_cell_id
No help available

6.1.1.4.9 Pdynamics

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
```

class PdynamicsCls

Pdynamics commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_tmask() → TimeMask

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
value: enums.TimeMask = driver.configure.lteMeas.multiEval.pdynamics.get_tmask()
```

Selects the time mask for power dynamics measurements.

return
time_mask: GOO: General time mask PPSRs: PUCCH/PUSCH transmission before and after an SRS SBLanking: SRS blanking time mask

set_tmask(*time_mask: TimeMask*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
driver.configure.lteMeas.multiEval.pdynamics.set_tmask(time_mask = enums.
↪TimeMask.GOO)
```

Selects the time mask for power dynamics measurements.

param time_mask
GOO: General time mask PPSRs: PUCCH/PUSCH transmission before and after an SRS SBLanking: SRS blanking time mask

Cloning the Group

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

Subgroups

6.1.1.4.9.1 AeoPower

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AEOPower:LEADing
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AEOPower:LAGGing
```

class AeoPowerCls

AeoPower commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_lagging() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LAGGing
value: int = driver.configure.lteMeas.multiEval.pdynamics.aeoPower.get_lagging()
```

Shifts the end of the evaluation period for OFF power measurements.

return
lagging: Positive values reduce the evaluation period (ends earlier) . Negative values increase the evaluation period (ends later) .

get_leading() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LEADing
value: int = driver.configure.lteMeas.multiEval.pdynamics.aeoPower.get_leading()
```

Shifts the beginning of the evaluation period for OFF power measurements.

return
leading: Positive values reduce the evaluation period (starts later) . Negative values increase the evaluation period (starts earlier) .

set_lagging(*lagging: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LAGGing
driver.configure.lteMeas.multiEval.pdynamics.aeoPower.set_lagging(lagging = 1)
```

Shifts the end of the evaluation period for OFF power measurements.

param lagging
Positive values reduce the evaluation period (ends earlier) . Negative values increase the evaluation period (ends later) .

set_leading(*leading: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LEADing
driver.configure.lteMeas.multiEval.pdynamics.aeoPower.set_leading(leading = 1)
```

Shifts the beginning of the evaluation period for OFF power measurements.

param leading
Positive values reduce the evaluation period (starts later) . Negative values increase the evaluation period (starts earlier) .

6.1.1.4.10 Power

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:POWer:HDMode
```

class PowerCls

Power commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_hdmode() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:POWer:HDMode
value: bool = driver.configure.lteMeas.multiEval.power.get_hdmode()
```

Enables or disables the high dynamic mode for power dynamics measurements.

return
high_dynamic_mode: No help available

set_hdmode(*high_dynamic_mode: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:POWer:HDMode
driver.configure.lteMeas.multiEval.power.set_hdmode(high_dynamic_mode = False)
```

Enables or disables the high dynamic mode for power dynamics measurements.

param high_dynamic_mode
No help available

6.1.1.4.11 RbAllocation

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
```

class RbAllocationCls

RbAllocation commands group definition. 10 total commands, 3 Subgroups, 1 group commands

get_auto() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
value: bool = driver.configure.lteMeas.multiEval.rbAllocation.get_auto()
```

Enables or disables the automatic detection of the RB configuration.

return
auto: OFF: manual definition ON: automatic detection

set_auto(*auto: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
driver.configure.lteMeas.multiEval.rbAllocation.set_auto(auto = False)
```

Enables or disables the automatic detection of the RB configuration.

param auto
OFF: manual definition ON: automatic detection

Cloning the Group

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

Subgroups

6.1.1.4.11.1 Mcluster

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster
```

class MclusterCls

Mcluster commands group definition. 3 total commands, 2 Subgroups, 1 group commands

get_value() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster
value: bool = driver.configure.lteMeas.multiEval.rbAllocation.mcluster.get_
↪value()
```

Specifies whether the UL signal uses multi-cluster allocation or not.

return
enable: OFF: contiguous allocation, resource allocation type 0 ON: multi-cluster allocation, resource allocation type 1

set_value(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster
driver.configure.lteMeas.multiEval.rbAllocation.mcluster.set_value(enable =␣
↪False)
```

Specifies whether the UL signal uses multi-cluster allocation or not.

param enable
OFF: contiguous allocation, resource allocation type 0 ON: multi-cluster allocation, resource allocation type 1

Cloning the Group

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

Subgroups

6.1.1.4.11.2 Nrb<RBcount>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.lteMeas.multiEval.rbAllocation.mcluster.nrb.repcap_rBcount_get()
driver.configure.lteMeas.multiEval.rbAllocation.mcluster.nrb.repcap_rBcount_set(repcap.
↪RBcount.Nr1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster:NRB<Number>
```

class NrbCls

Nrb commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: RBcount, default value after init: RBcount.Nr1

get(*rBcount=RBcount.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:NRB<Number>
value: int = driver.configure.lteMeas.multiEval.rbAllocation.mcluster.nrb.
↪get(rBcount = repcap.RBcount.Default)
```

Specifies the number of allocated RBs in the measured slot, for multi-cluster allocation.

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

return
no_rb: For the allowed input ranges, see 'Uplink resource block allocation'.

set(*no_rb: int*, *rBcount=RBcount.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:NRB<Number>
driver.configure.lteMeas.multiEval.rbAllocation.mcluster.nrb.set(no_rb = 1,␣
↪rBcount = repcap.RBcount.Default)
```

Specifies the number of allocated RBs in the measured slot, for multi-cluster allocation.

param no_rb
For the allowed input ranges, see 'Uplink resource block allocation'.

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

Cloning the Group

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

6.1.1.4.11.3 Orb<RBoffset>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.lteMeas.multiEval.rbAllocation.mcluster.orb.repcap_rBoffset_get()
driver.configure.lteMeas.multiEval.rbAllocation.mcluster.orb.repcap_rBoffset_set(repcap.
↪RBoffset.Nr1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:MCLuster:ORB<Number>
```

class OrbCls

Orb commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: RBoffset, default value after init: RBoffset.Nr1

get(*rBoffset=RBoffset.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:ORB<Number>
value: int = driver.configure.lteMeas.multiEval.rbAllocation.mcluster.orb.
↪get(rBoffset = repcap.RBoffset.Default)
```

Specifies the offset of the first allocated resource block, for multi-cluster allocation.

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

return
offset_rb: For the allowed input ranges, see 'Uplink resource block allocation'.

set(*offset_rb: int*, *rBoffset=RBoffset.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RBALlocation:MCLuster:ORB<Number>
driver.configure.lteMeas.multiEval.rbAllocation.mcluster.orb.set(offset_rb = 1,␣
↪rBoffset = repcap.RBoffset.Default)
```

Specifies the offset of the first allocated resource block, for multi-cluster allocation.

param offset_rb
For the allowed input ranges, see 'Uplink resource block allocation'.

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

Cloning the Group

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

6.1.1.4.11.4 Nrb

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSCCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSSCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB
```

class NrbCls

Nrb commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_pscch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSCCh
value: int = driver.configure.lteMeas.multiEval.rbAllocation.nrb.get_pscch()
```

Specifies the number of allocated RBs for the PSCCH in the measured slot. For manual RB allocation definition, for sidelink signals.

return
no_rb: The value is fixed.

get_pssch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSSCh
value: int = driver.configure.lteMeas.multiEval.rbAllocation.nrb.get_pssch()
```

Specifies the number of allocated RBs for the PSSCH in the measured slot. For manual RB allocation definition, for sidelink signals.

return
no_rb: For the allowed input range, see 'Sidelink resource block allocation'.

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB
value: int = driver.configure.lteMeas.multiEval.rbAllocation.nrb.get_value()
```

Specifies the number of allocated RBs in the measured slot. For manual RB allocation definition, for uplink signals without multi-cluster allocation.

return
no_rb: For the allowed input range, see 'Uplink resource block allocation'.

set_pscch(*no_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSCCh
driver.configure.lteMeas.multiEval.rbAllocation.nrb.set_pscch(no_rb = 1)
```

Specifies the number of allocated RBs for the PSCCH in the measured slot. For manual RB allocation definition, for sidelink signals.

param no_rb
The value is fixed.

set_pssch(*no_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB:PSSCh
driver.configure.lteMeas.multiEval.rbAllocation.nrb.set_pssch(no_rb = 1)
```

Specifies the number of allocated RBs for the PSSCH in the measured slot. For manual RB allocation definition, for sidelink signals.

param no_rb
For the allowed input range, see 'Sidelink resource block allocation'.

set_value(*no_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:NRB
driver.configure.lteMeas.multiEval.rbAllocation.nrb.set_value(no_rb = 1)
```

Specifies the number of allocated RBs in the measured slot. For manual RB allocation definition, for uplink signals without multi-cluster allocation.

param no_rb
For the allowed input range, see 'Uplink resource block allocation'.

6.1.1.4.11.5 Orb

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSCCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSSCh
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB
```

class OrbCls

Orb commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_pscch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSCCh
value: int = driver.configure.lteMeas.multiEval.rbAllocation.orb.get_pscch()
```

Specifies the offset of the first allocated PSCCH resource block for manual RB allocation definition, for sidelink signals.

return
offset_rb: For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'.

get_pssch() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSSCh
value: int = driver.configure.lteMeas.multiEval.rbAllocation.orb.get_pssch()
```

Specifies the offset of the first allocated PSSCH resource block for manual RB allocation definition, for sidelink signals.

return
offset_rb: The range depends on the OffsetRB for the PSCCH, the channel BW and the number of allocated PSSCH RBs, see 'Sidelink resource block allocation'.

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB
value: int = driver.configure.lteMeas.multiEval.rbAllocation.orb.get_value()
```

Specifies the offset of the first allocated resource block for manual RB allocation definition, for uplink signals without multi-cluster allocation.

return
offset_rb: For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'.

set_pscch(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSCCh
driver.configure.lteMeas.multiEval.rbAllocation.orb.set_pscch(offset_rb = 1)
```

Specifies the offset of the first allocated PSCCH resource block for manual RB allocation definition, for sidelink signals.

param offset_rb
For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'.

set_pssch(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB:PSSCh
driver.configure.lteMeas.multiEval.rbAllocation.orb.set_pssch(offset_rb = 1)
```

Specifies the offset of the first allocated PSSCH resource block for manual RB allocation definition, for sidelink signals.

param offset_rb
The range depends on the OffsetRB for the PSCCH, the channel BW and the number of allocated PSSCH RBs, see 'Sidelink resource block allocation'.

set_value(*offset_rb: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocation:ORB
driver.configure.lteMeas.multiEval.rbAllocation.orb.set_value(offset_rb = 1)
```

Specifies the offset of the first allocated resource block for manual RB allocation definition, for uplink signals without multi-cluster allocation.

param offset_rb
For the maximum number of RBs depending on the channel BW, see 'Uplink resource block allocation'.

6.1.1.4.12 Result

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult[:ALL]
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:MERRor
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PERRor
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IEMissions
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMC
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ESFLatness
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:TXM
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IQ
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:SEMask
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ACLR
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:RBATable
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PMONitor
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PDYNamics
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:BLER
```

class ResultCls

Result commands group definition. 16 total commands, 1 Subgroups, 14 group commands

class AllStruct

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

- Evm: bool: Error vector magnitude OFF: Do not evaluate the results. ON: Evaluate the results.
- Magnitude_Error: bool: No parameter help available
- Phase_Error: bool: No parameter help available
- Inband_Emissions: bool: No parameter help available
- Evm_Versus_C: bool: EVM vs subcarrier
- Iq: bool: I/Q constellation diagram
- Equ_Spec_Flatness: bool: Equalizer spectrum flatness
- Tx_Measurement: bool: TX measurement statistical overview
- Spec_Em_Mask: bool: Spectrum emission mask
- Aclr: bool: Adjacent channel leakage power ratio
- Rb_Alloc_Table: bool: Optional setting parameter. Resource block allocation table
- Power_Monitor: bool: No parameter help available
- Bler: bool: Optional setting parameter. Not supported
- Power_Dynamics: bool: No parameter help available

get_aclr() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ACLR
value: bool = driver.configure.lteMeas.multiEval.result.get_aclr()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.

Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_all() → AllStruct

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

Enables or disables the evaluation of results in the multi-evaluation measurement. This command combines most other CONFigure:LTE:MEAS<i>:MEValuation:RESult... commands.

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

get_bler() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:BLER
value: bool = driver.configure.lteMeas.multiEval.result.get_bler()
```

No command help available

return
enable: No help available

get_es_flatness() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ESFLatness
value: bool = driver.configure.lteMeas.multiEval.result.get_es_flatness()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude

- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_evmc() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMC
value: bool = driver.configure.lteMeas.multiEval.result.get_evmc()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_iemissions() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IEMissions
value: bool = driver.configure.lteMeas.multiEval.result.get_iemissions()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_iq() → bool

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

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table

-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_merror() → bool

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

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_pdynamics() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PDYNamics
value: bool = driver.configure.lteMeas.multiEval.result.get_pdynamics()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_perror() → bool

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

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier

- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_pmonitor() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PMONitor
value: bool = driver.configure.lteMeas.multiEval.result.get_pmonitor()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_rba_table() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:RBATable
value: bool = driver.configure.lteMeas.multiEval.result.get_rba_table()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_se_mask() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:SEMask
value: bool = driver.configure.lteMeas.multiEval.result.get_se_mask()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio

- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_txm() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:TXM
value: bool = driver.configure.lteMeas.multiEval.result.get_txm()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

set_aclr(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ACLR
driver.configure.lteMeas.multiEval.result.set_aclr(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions

- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_all(*value: AllStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult[:ALL]
structure = driver.configure.lteMeas.multiEval.result.AllStruct()
structure.Evm: bool = False
structure.Magnitude_Error: bool = False
structure.Phase_Error: bool = False
structure.Inband_Emissions: bool = False
structure.Evm_Versus_C: bool = False
structure.Iq: bool = False
structure.Equ_Spec_Flatness: bool = False
structure.Tx_Measurement: bool = False
structure.Spec_Em_Mask: bool = False
structure.Aclr: bool = False
structure.Rb_Alloc_Table: bool = False
structure.Power_Monitor: bool = False
structure.Bler: bool = False
structure.Power_Dynamics: bool = False
driver.configure.lteMeas.multiEval.result.set_all(value = structure)
```

Enables or disables the evaluation of results in the multi-evaluation measurement. This command combines most other CONFigure:LTE:MEAS<i>:MEValuation:RESult... commands.

param value
see the help for AllStruct structure arguments.

set_bler(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:BLER
driver.configure.lteMeas.multiEval.result.set_bler(enable = False)
```

No command help available

param enable
No help available

`set_es_flatness`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:ESFLatness
driver.configure.lteMeas.multiEval.result.set_es_flatness(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

`set_evmc`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMC
driver.configure.lteMeas.multiEval.result.set_evmc(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-

- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_iemissions(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:IEMissions
driver.configure.lteMeas.multiEval.result.set_iemissions(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_iq(*enable: bool*) → None

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

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

`set_merror`(*enable: bool*) → None

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

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio

- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_pdynamics(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PDYNamics
driver.configure.lteMeas.multiEval.result.set_pdynamics(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_perror(*enable: bool*) → None

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

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions

- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_pmonitor(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:PMONitor
driver.configure.lteMeas.multiEval.result.set_pmonitor(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_rba_table(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:RBATable
driver.configure.lteMeas.multiEval.result.set_rba_table(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_se_mask(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:SEMask
driver.configure.lteMeas.multiEval.result.set_se_mask(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier

- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_txm(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:TXM
driver.configure.lteMeas.multiEval.result.set_txm(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

Cloning the Group

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

Subgroups

6.1.1.4.12.1 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
```

class EvMagnitudeCls

EvMagnitude commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_value() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
value: bool = driver.configure.lteMeas.multiEval.result.evMagnitude.get_value()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.

Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

return

enable: OFF: Do not evaluate the results. ON: Evaluate the results.

set_value(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
driver.configure.lteMeas.multiEval.result.evMagnitude.set_value(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- IEMissions / In-band emissions
- ESFLatness / Equalizer spectrum flatness
- SEMask / Spectrum emission mask
- RBATable / Resource block allocation table
-
- EVMC / EVM vs subcarrier
- PERRor / Phase error
- IQ / I/Q constellation diagram
- TXM / TX meas. statistical overview
- ACLR / Adj. channel leakage power ratio
- PMONitor / Power monitor
- PDYNamics / Power dynamics

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

Cloning the Group

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

Subgroups

6.1.1.4.12.2 EvmSymbol

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:RESult:EVMagnitude:EVMSymbol
```

class EvmSymbolCls

EvmSymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EvmSymbolStruct

Response structure. Fields:

- Enable: bool: OFF: Do not measure the results. ON: Measure the results.
- Symbol: int: SC-FDMA symbol to be evaluated.
- Low_High: enums.LowHigh: Low or high EVM window position.

get() → EvmSymbolStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RESult:EVMagnitude:EVMSymbol
value: EvmSymbolStruct = driver.configure.lteMeas.multiEval.result.evMagnitude.
↪evmSymbol.get()
```

Enables or disables the measurement of EVM vs modulation symbol results and configures the scope of the measurement.

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

set(*enable: bool*, *symbol: int*, *low_high: LowHigh*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>
↪:MEValuation:RESult:EVMagnitude:EVMSymbol
driver.configure.lteMeas.multiEval.result.evMagnitude.evmSymbol.set(enable =␣
↪False, symbol = 1, low_high = enums.LowHigh.HIGH)
```

Enables or disables the measurement of EVM vs modulation symbol results and configures the scope of the measurement.

param enable
OFF: Do not measure the results. ON: Measure the results.

param symbol
SC-FDMA symbol to be evaluated.

param low_high
Low or high EVM window position.

6.1.1.4.13 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.lteMeas.multiEval.scc.repcap_secondaryCC_get()
driver.configure.lteMeas.multiEval.scc.repcap_secondaryCC_set(repcap.SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.1.1.4.13.1 PlcId

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCC<Nr>:PLCid
```

class PlcIdCls

PlcId commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*secondaryCC=SecondaryCC.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCC<Nr>:PLCid
value: int = driver.configure.lteMeas.multiEval.scc.plcId.get(secondaryCC =␣
↪repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
phs_layer_cell_id: No help available

set(*phs_layer_cell_id: int*, *secondaryCC=SecondaryCC.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCC<Nr>:PLCid
driver.configure.lteMeas.multiEval.scc.plcId.set(phs_layer_cell_id = 1,␣
↪secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param phs_layer_cell_id
No help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

6.1.1.4.14 Scount

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:POWer
```

class ScountCls

Scount commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_modulation() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:MODulation
value: int = driver.configure.lteMeas.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: No help available

get_power() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:POWer
value: int = driver.configure.lteMeas.multiEval.scount.get_power()
```

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

return
statistic_count: No help available

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:MODulation
driver.configure.lteMeas.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
No help available

set_power(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:POWer
driver.configure.lteMeas.multiEval.scount.set_power(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
No help available

Cloning the Group

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

Subgroups

6.1.1.4.14.1 Spectrum

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
```

class SpectrumCls

Spectrum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_aclr() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
value: int = driver.configure.lteMeas.multiEval.scount.spectrum.get_aclr()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

return
statistic_count: No help available

get_se_mask() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
value: int = driver.configure.lteMeas.multiEval.scount.spectrum.get_se_mask()
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

return
statistic_count: No help available

set_aclr(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
driver.configure.lteMeas.multiEval.scount.spectrum.set_aclr(statistic_count = 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

param statistic_count
No help available

set_se_mask(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
driver.configure.lteMeas.multiEval.scount.spectrum.set_se_mask(statistic_count
↪= 1)
```

Specifies the statistic count of the measurement. The statistic count is equal to the number of measurement intervals per single shot. Separate statistic counts for ACLR and spectrum emission mask measurements are supported.

param statistic_count
No help available

6.1.1.4.15 Spectrum

class SpectrumCls

Spectrum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.15.1 Aclr

class AclrCls

Aclr commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.4.15.2 Enable

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
```

class EnableCls

Enable commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EnableStruct

Response structure. Fields:

- Utra_1: bool: No parameter help available
- Utra_2: bool: No parameter help available
- Eutra: bool: No parameter help available

get() → EnableStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
value: EnableStruct = driver.configure.lteMeas.multiEval.spectrum.aclr.enable.
↪get()
```

Enables or disables the evaluation of the first adjacent UTRA channels, second adjacent UTRA channels and first adjacent E-UTRA channels.

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

set(*utra_1: bool*, *utra_2: bool*, *eutra: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
driver.configure.lteMeas.multiEval.spectrum.aclr.enable.set(utra_1 = False,␣
↪utra_2 = False, eutra = False)
```

Enables or disables the evaluation of the first adjacent UTRA channels, second adjacent UTRA channels and first adjacent E-UTRA channels.

param utra_1
No help available

param utra_2
No help available

param eutra
No help available

6.1.1.4.15.3 SeMask

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:MFILter
```

class SeMaskCls

SeMask commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_mfilter() → MeasFilter

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:MFILter
value: enums.MeasFilter = driver.configure.lteMeas.multiEval.spectrum.seMask.
↪get_mfilter()
```

Selects the resolution filter type for filter bandwidths of 50 kHz and greater.

return
meas_filter: No help available

set_mfilter(*meas_filter: MeasFilter*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:MFILter
driver.configure.lteMeas.multiEval.spectrum.seMask.set_mfilter(meas_filter =␣
↪enums.MeasFilter.BANDpass)
```

Selects the resolution filter type for filter bandwidths of 50 kHz and greater.

param meas_filter
No help available

6.1.1.4.16 Srs

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:SRS:ENABle
```

class SrsCls

Srs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_enable() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SRS:ENABle
value: bool = driver.configure.lteMeas.multiEval.srs.get_enable()
```

Specifies whether a sounding reference signal is allowed (ON) or not (OFF) . For Signal Path = Network, the setting is not configurable.

return
enable: OFF: No SRS signal is allowed. ON: An SRS signal is allowed in the last SC-FDMA symbol of each subframe.

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:SRS:ENABle
driver.configure.lteMeas.multiEval.srs.set_enable(enable = False)
```

Specifies whether a sounding reference signal is allowed (ON) or not (OFF) . For Signal Path = Network, the setting is not configurable.

param enable
OFF: No SRS signal is allowed. ON: An SRS signal is allowed in the last SC-FDMA symbol of each subframe.

6.1.1.4.17 Tmode

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:SCOunt
CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:ENPower
CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:RLEVel
```

class TmodeCls

Tmode commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_envelope_power() → List[float]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:ENPower
value: List[float] = driver.configure.lteMeas.multiEval.tmode.get_envelope_
↪power()
```

Defines the expected nominal power values for all entries of the TPC Mode list. For the definition of the corresponding subframe count values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Tmode.scount.

return
exp_nom_pow: Comma-separated list of 16 values, for list entry number 0 to 15 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 specifications document.

get_rlevel() → List[float]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:RLEVel
value: List[float] = driver.configure.lteMeas.multiEval.tmode.get_rlevel()
```

Queries the reference level for all entries of the TPC Mode list. The reference level is calculated from the expected nominal power of each entry and the user margin.

return
reference_level: Comma-separated list of 16 values, for list entry number 0 to 15 The range of the reference levels can be calculated as follows: Range (Reference Level) = Range (Input Power) + External Attenuation The input power range is stated in the specifications document.

get_scount() → List[int]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:SCOunt
value: List[int] = driver.configure.lteMeas.multiEval.tmode.get_scount()
```

Defines the subframe counts for all entries of the TPC Mode list. For the definition of the corresponding expected nominal power values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Tmode.envelopePower.

return
sub_frame_count: Comma-separated list of 16 values, for list entry number 0 to 15

set_envelope_power(*exp_nom_pow: List[float]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:ENPower
driver.configure.lteMeas.multiEval.tmode.set_envelope_power(exp_nom_pow = [1.1,␣
↪2.2, 3.3])
```

Defines the expected nominal power values for all entries of the TPC Mode list. For the definition of the corresponding subframe count values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Tmode.scount.

param exp_nom_pow
Comma-separated list of 16 values, for list entry number 0 to 15 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 specifications document.

set_scount(*sub_frame_count: List[int]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:MEValuation:TMODe:SCOunt
driver.configure.lteMeas.multiEval.tmode.set_scount(sub_frame_count = [1, 2, 3])
```

Defines the subframe counts for all entries of the TPC Mode list. For the definition of the corresponding expected nominal power values, see method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.Tmode.envelopePower.

param sub_frame_count
Comma-separated list of 16 values, for list entry number 0 to 15

6.1.1.5 Network

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:NETWork:RFPSharing
CONFigure:LTE:MEASurement<Instance>:NETWork:BAND
CONFigure:LTE:MEASurement<Instance>:NETWork:DMODe
```

class NetworkCls

Network commands group definition. 3 total commands, 0 Subgroups, 3 group commands

get_band() → Band

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

No command help available

return
band: No help available

get_dmode() → Mode

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:DMODe
value: enums.Mode = driver.configure.lteMeas.network.get_dmode()
```

No command help available

return
mode: No help available

get_rfp_sharing() → NetworkSharing

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:RFPSharing
value: enums.NetworkSharing = driver.configure.lteMeas.network.get_rfp_sharing()
```

Selects the RF path sharing mode for a measurement with coupling to signaling settings.

return
sharing: NSHared: not shared (independent connection) OCONnection: only connection shared FSHared: fully shared (only for RF unit)

set_band(*band: Band*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:BAND
driver.configure.lteMeas.network.set_band(band = enums.Band.OB1)
```

No command help available

param band
No help available

set_dmode(*mode: Mode*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:DMODe
driver.configure.lteMeas.network.set_dmode(mode = enums.Mode.FDD)
```

No command help available

param mode
No help available

set_rfp_sharing(*sharing: NetworkSharing*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:NETWork:RFPSharing
driver.configure.lteMeas.network.set_rfp_sharing(sharing = enums.NetworkSharing.
↪FSHared)
```

Selects the RF path sharing mode for a measurement with coupling to signaling settings.

param sharing
NSHared: not shared (independent connection) OCONnection: only connection shared FSHared: fully shared (only for RF unit)

6.1.1.6 Pcc

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>[:PCC]:CBANdwidth
```

class PccCls

Pcc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_channel_bw() → ChannelBandwidth

```
# SCPI: CONFigure:LTE:MEASurement<Instance>[:PCC]:CBANdwidth
value: enums.ChannelBandwidth = driver.configure.lteMeas.pcc.get_channel_bw()
```

No command help available

return
channel_bw: No help available

set_channel_bw(*channel_bw: ChannelBandwidth*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>[:PCC]:CBANdwidth
driver.configure.lteMeas.pcc.set_channel_bw(channel_bw = enums.ChannelBandwidth.
↪B014)
```

No command help available

param channel_bw
No help available

6.1.1.7 Prach

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:VIEW
CONFigure:LTE:MEASurement<Instance>:PRACh:TOUT
CONFigure:LTE:MEASurement<Instance>:PRACh:REPetition
CONFigure:LTE:MEASurement<Instance>:PRACh:SCONdition
CONFigure:LTE:MEASurement<Instance>:PRACh:MOEXception
CONFigure:LTE:MEASurement<Instance>:PRACh:PCINdex
CONFigure:LTE:MEASurement<Instance>:PRACh:SSYMbol
CONFigure:LTE:MEASurement<Instance>:PRACh:PFORmat
CONFigure:LTE:MEASurement<Instance>:PRACh:NOPReambles
CONFigure:LTE:MEASurement<Instance>:PRACh:POPReambles
```

class PrachCls

Prach commands group definition. 36 total commands, 6 Subgroups, 10 group commands

get_mo_exception() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MOEXception
value: bool = driver.configure.lteMeas.prach.get_mo_exception()
```

Specifies whether measurement results that the CMX500 identifies as faulty or inaccurate are rejected.

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

get_no_preambles() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:NOPReambles
value: int = driver.configure.lteMeas.prach.get_no_preambles()
```

Specifies the number of preambles to be captured per measurement interval.

return
number_preamble: No help available

get_pc_index() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PCINdex
value: int = driver.configure.lteMeas.prach.get_pc_index()
```

The PRACH configuration index identifies the PRACH configuration used by the UE (preamble format, which resources in the time domain are allowed for transmission of preambles etc.) . For Signal Path = Network, use[CONFigure:]SIGNaling:LTE:CELL:POWer:UL:CINDex.

return
prach_conf_index: No help available

get_pformat() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFORmat
value: int = driver.configure.lteMeas.prach.get_pformat()
```

No command help available

return
preamble_format: No help available

get_po_preambles() → PeriodPreamble

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POPReambles
value: enums.PeriodPreamble = driver.configure.lteMeas.prach.get_po_preambles()
```

Specifies the periodicity of preambles to be captured for multi-preamble result views.

return
period_preamble: MS05: 5 ms MS10: 10 ms MS20: 20 ms

get_repetition() → Repeat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:REPetition
value: enums.Repeat = driver.configure.lteMeas.prach.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: Single-shot measurement CONTinuous: Continuous measurement

get_scondition() → StopCondition

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCONdition
value: enums.StopCondition = driver.configure.lteMeas.prach.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: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

get_ssymbol() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SSYMbol
value: int = driver.configure.lteMeas.prach.get_ssymbol()
```

Selects the OFDM symbol to be evaluated for single-symbol modulation result diagrams. The number of OFDM symbols in the preamble (<no of symbols>) depends on the preamble format, see Table 'Preambles in the time domain'.

return
selected_symbol: No help available

get_timeout() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:TOUT
value: float = driver.configure.lteMeas.prach.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. 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: No help available

get_view() → ViewPrach

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:VIEW
value: enums.ViewPrach = driver.configure.lteMeas.prach.get_view()
```

No command help available

return
view: No help available

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MOEXception
driver.configure.lteMeas.prach.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the CMX500 identifies as faulty or inaccurate are rejected.

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

set_no_preambles(*number_preamble: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:NOPReambles
driver.configure.lteMeas.prach.set_no_preambles(number_preamble = 1)
```

Specifies the number of preambles to be captured per measurement interval.

param number_preamble
No help available

set_pc_index(*prach_conf_index: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PCINdex
driver.configure.lteMeas.prach.set_pc_index(prach_conf_index = 1)
```

The PRACH configuration index identifies the PRACH configuration used by the UE (preamble format, which resources in the time domain are allowed for transmission of preambles etc.) . For Signal Path = Network, use[CONFigure:]SIGNaling:LTE:CELL:POWer:UL:CINDex.

param prach_conf_index
No help available

set_po_preambles(*period_preamble: PeriodPreamble*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POPReambles
driver.configure.lteMeas.prach.set_po_preambles(period_preamble = enums.
↪PeriodPreamble.MS05)
```

Specifies the periodicity of preambles to be captured for multi-preamble result views.

param period_preamble
MS05: 5 ms MS10: 10 ms MS20: 20 ms

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:REPetition
driver.configure.lteMeas.prach.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: Single-shot measurement CONTinuous: Continuous measurement

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCONdition
driver.configure.lteMeas.prach.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: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

set_ssymbol(*selected_symbol: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SSYMbol
driver.configure.lteMeas.prach.set_ssymbol(selected_symbol = 1)
```

Selects the OFDM symbol to be evaluated for single-symbol modulation result diagrams. The number of OFDM symbols in the preamble (<no of symbols>) depends on the preamble format, see Table 'Preambles in the time domain'.

param selected_symbol
No help available

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:TOUT
driver.configure.lteMeas.prach.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. 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
No help available

set_view(*view: ViewPrach*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:VIEW
driver.configure.lteMeas.prach.set_view(view = enums.ViewPrach.EVMagnitude)
```

No command help available

param view
No help available

Cloning the Group

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

Subgroups

6.1.1.7.1 Limit

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:FERRor
```

class LimitCls

Limit commands group definition. 5 total commands, 4 Subgroups, 1 group commands

get_freq_error() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:FERRor
value: float or bool = driver.configure.lteMeas.prach.limit.get_freq_error()
```

Defines an upper limit for the carrier frequency error.

return
frequency_error: (float or boolean) No help available

set_freq_error(*frequency_error: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:FERRor
driver.configure.lteMeas.prach.limit.set_freq_error(frequency_error = 1.0)
```

Defines an upper limit for the carrier frequency error.

param frequency_error
(float or boolean) No help available

Cloning the Group

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

Subgroups

6.1.1.7.1.1 EvMagnitude

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
```

class EvMagnitudeCls

EvMagnitude commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class EvMagnitudeStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get() → EvMagnitudeStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
value: EvMagnitudeStruct = driver.configure.lteMeas.prach.limit.evMagnitude.
↪get()
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) .

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
driver.configure.lteMeas.prach.limit.evMagnitude.set(rms = 1.0, peak = 1.0)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) .

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.7.1.2 Merror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:MERRor
```

class MerrorCls

Merror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class MerrorStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get() → MerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:MERRor
value: MerrorStruct = driver.configure.lteMeas.prach.limit.merror.get()
```

Defines upper limits for the RMS and peak values of the magnitude error.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:MERRor
driver.configure.lteMeas.prach.limit.merror.set(rms = 1.0, peak = 1.0)
```

Defines upper limits for the RMS and peak values of the magnitude error.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.7.1.3 Pdynamics

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PDYNamics
```

class PdynamicsCls

Pdynamics commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PdynamicsStruct

Response structure. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- On_Power_Upper: float: Upper limit for the ON power
- On_Power_Lower: float: Lower limit for the ON power

- Off_Power_Upper: float: Upper limit for the OFF power

get() → PdynamicsStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PDYNamics
value: PdynamicsStruct = driver.configure.lteMeas.prach.limit.pdynamics.get()
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

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

set(*enable: bool*, *on_power_upper: float*, *on_power_lower: float*, *off_power_upper: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PDYNamics
driver.configure.lteMeas.prach.limit.pdynamics.set(enable = False, on_power_
↪upper = 1.0, on_power_lower = 1.0, off_power_upper = 1.0)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

param enable
OFF: disables the limit check ON: enables the limit check

param on_power_upper
Upper limit for the ON power

param on_power_lower
Lower limit for the ON power

param off_power_upper
Upper limit for the OFF power

6.1.1.7.1.4 Perror

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PERRor
```

class PerrorCls

Perror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PerrorStruct

Response structure. Fields:

- Rms: float or bool: No parameter help available
- Peak: float or bool: No parameter help available

get() → PerrorStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PERRor
value: PerrorStruct = driver.configure.lteMeas.prach.limit.perror.get()
```

Defines symmetric limits for the RMS and peak values of the phase error. The limit check fails if the absolute value of the measured phase error exceeds the specified limit.

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

set(*rms: float*, *peak: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:LIMit:PERRor
driver.configure.lteMeas.prach.limit.perror.set(rms = 1.0, peak = 1.0)
```

Defines symmetric limits for the RMS and peak values of the phase error. The limit check fails if the absolute value of the measured phase error exceeds the specified limit.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.7.2 Modulation

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:LRSindex
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:ZCZConfig
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWPosition
```

class ModulationCls

Modulation commands group definition. 7 total commands, 2 Subgroups, 3 group commands

get_ew_position() → LowHigh

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWPosition
value: enums.LowHigh = driver.configure.lteMeas.prach.modulation.get_ew_
↪position()
```

Specifies the position of the EVM window used for calculation of the trace results.

return
evm_window_pos: No help available

get_lrs_index() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:LRSindex
value: int = driver.configure.lteMeas.prach.modulation.get_lrs_index()
```

Specifies the logical root sequence index to be used for generation of the preamble sequence. For Signal Path = Network, the setting is not configurable.

return
log_root_seq_index: No help available

get_zcz_config() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:ZCZConfig
value: int = driver.configure.lteMeas.prach.modulation.get_zcz_config()
```

Specifies the zero correlation zone config, i.e. which NCS value of an NCS set is used for generation of the preamble sequence. For Signal Path = Network, the setting is not configurable.

return
zero_corr_zone_con: No help available

set_ew_position(*evm_window_pos: LowHigh*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWPosition
driver.configure.lteMeas.prach.modulation.set_ew_position(evm_window_pos =␣
↪enums.LowHigh.HIGH)
```

Specifies the position of the EVM window used for calculation of the trace results.

param evm_window_pos
No help available

set_lrs_index(*log_root_seq_index: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:LRSindex
driver.configure.lteMeas.prach.modulation.set_lrs_index(log_root_seq_index = 1)
```

Specifies the logical root sequence index to be used for generation of the preamble sequence. For Signal Path = Network, the setting is not configurable.

param log_root_seq_index
No help available

set_zcz_config(*zero_corr_zone_con: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:ZCZConfig
driver.configure.lteMeas.prach.modulation.set_zcz_config(zero_corr_zone_con = 1)
```

Specifies the zero correlation zone config, i.e. which NCS value of an NCS set is used for generation of the preamble sequence. For Signal Path = Network, the setting is not configurable.

param zero_corr_zone_con
No help available

Cloning the Group

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

Subgroups

6.1.1.7.2.1 EwLength

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength
```

class EwLengthCls

EwLength commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_value() → List[int]

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength
value: List[int] = driver.configure.lteMeas.prach.modulation.ewLength.get_
↪value()
```

Specifies the EVM window length in samples for all preamble formats.

return
evm_window_length: No help available

set_value(*evm_window_length: List[int]*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength
driver.configure.lteMeas.prach.modulation.ewLength.set_value(evm_window_length␣
↪= [1, 2, 3])
```

Specifies the EVM window length in samples for all preamble formats.

param evm_window_length
No help available

Cloning the Group

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

Subgroups

6.1.1.7.2.2 Pformat<PreambleFormat>

RepCap Settings

```
# Range: Fmt1 .. Fmt5
rc = driver.configure.lteMeas.prach.modulation.ewLength.pformat.repcap_preambleFormat_
↪get()
driver.configure.lteMeas.prach.modulation.ewLength.pformat.repcap_preambleFormat_
↪set(repcap.PreambleFormat.Fmt1)
```

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat<PreambleFormat>
```

class PformatCls

Pformat commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: PreambleFormat, default value after init: PreambleFormat.Fmt1

get(*preambleFormat=PreambleFormat.Default*) → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat
↪<PreambleFormat>
value: int = driver.configure.lteMeas.prach.modulation.ewLength.pformat.
↪get(preambleFormat = repcap.PreambleFormat.Default)
```

No command help available

param preambleFormat
optional repeated capability selector. Default value: Fmt1 (settable in the interface 'Pformat')

return
evm_window_length: No help available

set(*evm_window_length: int*, *preambleFormat=PreambleFormat.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat
↪<PreambleFormat>
driver.configure.lteMeas.prach.modulation.ewLength.pformat.set(evm_window_
↪length = 1, preambleFormat = repcap.PreambleFormat.Default)
```

No command help available

param evm_window_length
No help available

param preambleFormat
optional repeated capability selector. Default value: Fmt1 (settable in the interface 'Pformat')

Cloning the Group

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

6.1.1.7.2.3 Sindex

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex:AUTO
CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex
```

class SindexCls

Sindex commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_auto() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex:AUTO
value: bool = driver.configure.lteMeas.prach.modulation.sindex.get_auto()
```

Enables or disables automatic detection of the sequence index. To configure the index manually for disabled automatic detection, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Modulation.Sindex.value.

return
seq_index_auto: No help available

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex
value: int = driver.configure.lteMeas.prach.modulation.sindex.get_value()
```

Specifies the sequence index, i.e. which of the 64 preamble sequences of the cell is used by the UE. This setting is only relevant if automatic detection is disabled, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Modulation.Sindex.auto.

return
sequence_index: No help available

set_auto(*seq_index_auto: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex:AUTO
driver.configure.lteMeas.prach.modulation.sindex.set_auto(seq_index_auto =␣
↪False)
```

Enables or disables automatic detection of the sequence index. To configure the index manually for disabled automatic detection, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Modulation.Sindex.value.

param seq_index_auto
No help available

set_value(*sequence_index: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:MODulation:SINDex
driver.configure.lteMeas.prach.modulation.sindex.set_value(sequence_index = 1)
```

Specifies the sequence index, i.e. which of the 64 preamble sequences of the cell is used by the UE. This setting is only relevant if automatic detection is disabled, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Modulation.Sindex.auto.

param sequence_index
No help available

6.1.1.7.3 PfOffset

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset:AUTO
CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset
```

class PfOffsetCls

PfOffset commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_auto() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset:AUTO
value: bool = driver.configure.lteMeas.prach.pfOffset.get_auto()
```

Enables or disables automatic detection of the PRACH frequency offset. To configure the offset manually for disabled automatic detection, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.PfOffset.value.

return
prach_freq_auto: No help available

get_value() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset
value: int = driver.configure.lteMeas.prach.pfOffset.get_value()
```

Specifies the PRACH frequency offset. This setting is only relevant if automatic detection is disabled, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.PfOffset.auto.

return
prach_freq_offset: No help available

set_auto(*prach_freq_auto: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset:AUTO
driver.configure.lteMeas.prach.pfOffset.set_auto(prach_freq_auto = False)
```

Enables or disables automatic detection of the PRACH frequency offset. To configure the offset manually for disabled automatic detection, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.PfOffset.value.

param prach_freq_auto
No help available

set_value(*prach_freq_offset: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:PFOFfset
driver.configure.lteMeas.prach.pfOffset.set_value(prach_freq_offset = 1)
```

Specifies the PRACH frequency offset. This setting is only relevant if automatic detection is disabled, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.PfOffset.auto.

param prach_freq_offset
No help available

6.1.1.7.4 Power

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:POWer:HDMode
```

class PowerCls

Power commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_hdmode() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POWer:HDMode
value: bool = driver.configure.lteMeas.prach.power.get_hdmode()
```

Enables or disables the high dynamic mode for power dynamics measurements. With RF path sharing, this command is not applicable.

return
high_dynamic_mode: No help available

set_hdmode(*high_dynamic_mode: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:POWer:HDMode
driver.configure.lteMeas.prach.power.set_hdmode(high_dynamic_mode = False)
```

Enables or disables the high dynamic mode for power dynamics measurements. With RF path sharing, this command is not applicable.

param high_dynamic_mode
No help available

6.1.1.7.5 Result

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult[:ALL]
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVMagnitude
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVPReamble
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:MERRor
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PERRor
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:IQ
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PDYNamics
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PVPReamble
CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:TXM
```

class ResultCls

Result commands group definition. 9 total commands, 0 Subgroups, 9 group commands

class AllStruct

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

- Evm: bool: Error vector magnitude OFF: Do not evaluate the results. ON: Evaluate the results.
- Magnitude_Error: bool: No parameter help available
- Phase_Error: bool: No parameter help available
- Iq: bool: I/Q constellation diagram
- Power_Dynamics: bool: No parameter help available
- Tx_Measurement: bool: TX measurement statistical overview
- Evm_Vs_Preamble: bool: No parameter help available
- Power_Vs_Preamble: bool: No parameter help available

get_all() → AllStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult[:ALL]
value: AllStruct = driver.configure.lteMeas.prach.result.get_all()
```

Enables or disables the evaluation of results in the PRACH measurement. This command combines all other CONFigure:LTE:MEAS<i>:PRACh:RESult... commands.

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

get_ev_magnitude() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVMagnitude
value: bool = driver.configure.lteMeas.prach.result.get_ev_magnitude()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_ev_preamble() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVPReamble
value: bool = driver.configure.lteMeas.prach.result.get_ev_preamble()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_iq() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:IQ
value: bool = driver.configure.lteMeas.prach.result.get_iq()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error

- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_merror() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:MERRor
value: bool = driver.configure.lteMeas.prach.result.get_merror()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_pdynamics() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PDYNamics
value: bool = driver.configure.lteMeas.prach.result.get_pdynamics()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview

- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

`get_perror()` → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PERRor
value: bool = driver.configure.lteMeas.prach.result.get_perror()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

`get_pv_preamble()` → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PVPReamble
value: bool = driver.configure.lteMeas.prach.result.get_pv_preamble()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_txm() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:TXM
value: bool = driver.configure.lteMeas.prach.result.get_txm()
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

set_all(*value: AllStruct*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult[:ALL]
structure = driver.configure.lteMeas.prach.result.AllStruct()
structure.Evm: bool = False
structure.Magnitude_Error: bool = False
structure.Phase_Error: bool = False
structure.Iq: bool = False
structure.Power_Dynamics: bool = False
structure.Tx_Measurement: bool = False
structure.Evm_Vs_Preamble: bool = False
structure.Power_Vs_Preamble: bool = False
driver.configure.lteMeas.prach.result.set_all(value = structure)
```

Enables or disables the evaluation of results in the PRACH measurement. This command combines all other CONFigure:LTE:MEAS<i>:PRACh:RESult... commands.

param value
see the help for AllStruct structure arguments.

set_ev_magnitude(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVMagnitude
driver.configure.lteMeas.prach.result.set_ev_magnitude(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_ev_preamble(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:EVPReamble
driver.configure.lteMeas.prach.result.set_ev_preamble(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_iq(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:IQ
driver.configure.lteMeas.prach.result.set_iq(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error

- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_merror(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:MERRor
driver.configure.lteMeas.prach.result.set_merror(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_pdynamics(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PDYNamics
driver.configure.lteMeas.prach.result.set_pdynamics(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview

- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

`set_perror`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PERRor
driver.configure.lteMeas.prach.result.set_perror(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

`set_pv_preamble`(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:PVPReamble
driver.configure.lteMeas.prach.result.set_pv_preamble(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_txm(*enable: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:RESult:TXM
driver.configure.lteMeas.prach.result.set_txm(enable = False)
```

Enables or disables the evaluation of results in the PRACH measurement.
Table Header: Mnemonic / Description

- EVMagnitude / Error vector magnitude
- MERRor / Magnitude error
- PERRor / Phase error
- IQ / I/Q constellation diagram
- PDYNamics / Power dynamics
- TXM / TX meas. statistical overview
- EVPReamble / EVM vs preamble
- PVPReamble / Power vs preamble

For reset values, see method RsCMPX_LteMeas.Configure.LteMeas.Prach.Result.all.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

6.1.1.7.6 Scount

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:MODulation
CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
```

class ScountCls

Scount commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_modulation() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:MODulation
value: int = driver.configure.lteMeas.prach.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: No help available

get_pdynamics() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
value: int = driver.configure.lteMeas.prach.scount.get_pdynamics()
```

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

return
statistic_count: No help available

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:MODulation
driver.configure.lteMeas.prach.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
No help available

set_pdynamics(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
driver.configure.lteMeas.prach.scount.set_pdynamics(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
No help available

6.1.1.8 RfSettings

SCPI Commands :

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

class RfSettingsCls

RfSettings commands group definition. 8 total commands, 3 Subgroups, 5 group commands

get_eattenuation() → float

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

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector. With full RF path sharing, this command is not applicable.

return
rf_input_ext_att: No help available

get_envelope_power() → float

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

Sets the expected nominal power of the measured RF signal. With full RF path sharing, use the signaling commands controlling the uplink power.

return
exp_nom_pow: 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 specifications document.

get_foffset() → int

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

No command help available

return
offset: No help available

get_ml_offset() → float

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

No command help available

return
mix_lev_offset: No help available

get_umargin() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:UMARgin
value: float = driver.configure.lteMeas.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 specifications document. With full RF path sharing, this command is not applicable.

return
user_margin: No help available

set_eattenuation(*rf_input_ext_att: float*) → None

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

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector. With full RF path sharing, this command is not applicable.

param rf_input_ext_att
No help available

set_envelope_power(*exp_nom_pow: float*) → None

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

Sets the expected nominal power of the measured RF signal. With full RF path sharing, use the signaling commands controlling the uplink power.

param exp_nom_pow
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 specifications document.

set_foffset(*offset: int*) → None

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

No command help available

param offset
No help available

set_ml_offset(*mix_lev_offset: float*) → None

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

No command help available

param mix_lev_offset
No help available

set_umargin(*user_margin: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:UMARgin
driver.configure.lteMeas.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 specifications document. With full RF path sharing, this command is not applicable.

param user_margin
No help available

Cloning the Group

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

Subgroups

6.1.1.8.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.lteMeas.rfSettings.cc.repcap_carrierComponent_get()
driver.configure.lteMeas.rfSettings.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.1.1.8.1.1 Frequency

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:RFSettings:CC<Nr>:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:CC<Nr>:FREQuency
value: float = driver.configure.lteMeas.rfSettings.cc.frequency.
↪get(carrierComponent = repcap.CarrierComponent.Default)
```

Selects the center frequency of component carrier CC<no>. Without carrier aggregation, you can omit <no>. Using the unit CH, the frequency can be set via the channel number. The allowed channel number range depends on the operating band, see 'Frequency bands'.

INTRO_CMD_HELP: For Signal Path = Network, use:

- [CONFigure:]SIGNaling:LTE:CELL:RFSettings:UL:FREQuency
- [CONFigure:]SIGNaling:LTE:CELL:RFSettings:UL:EARFcn

For the supported frequency range, see 'Frequency ranges'.

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

return
analyzer_freq: No help available

set(*analyzer_freq: float*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:CC<Nr>:FREQuency
driver.configure.lteMeas.rfSettings.cc.frequency.set(analyzer_freq = 1.0,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Selects the center frequency of component carrier CC<no>. Without carrier aggregation, you can omit <no>. Using the unit CH, the frequency can be set via the channel number. The allowed channel number range depends on the operating band, see 'Frequency bands'.

INTRO_CMD_HELP: For Signal Path = Network, use:

- [CONFigure:]SIGNaling:LTE:CELL:RFSettings:UL:FREQuency
- [CONFigure:]SIGNaling:LTE:CELL:RFSettings:UL:EARFcn

For the supported frequency range, see 'Frequency ranges'.

param analyzer_freq
No help available

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

6.1.1.8.2 Pcc

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:RFSettings[:PCC]:FREQuency
```

class PccCls

Pcc commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_frequency() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings[:PCC]:FREQuency
value: float = driver.configure.lteMeas.rfSettings.pcc.get_frequency()
```

No command help available

return
analyzer_freq: No help available

set_frequency(*analyzer_freq: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings[:PCC]:FREQuency
driver.configure.lteMeas.rfSettings.pcc.set_frequency(analyzer_freq = 1.0)
```

No command help available

param analyzer_freq
No help available

6.1.1.8.3 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.lteMeas.rfSettings.scc.repcap_secondaryCC_get()
driver.configure.lteMeas.rfSettings.scc.repcap_secondaryCC_set(repcap.SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.1.1.8.3.1 Frequency

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:RFSettings:SCC<Nr>:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*secondaryCC=SecondaryCC.Default*) → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:SCC<Nr>:FREQuency
value: float = driver.configure.lteMeas.rfSettings.scc.frequency.
↪get(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
analyzer_freq: No help available

set(*analyzer_freq: float*, *secondaryCC=SecondaryCC.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:RFSettings:SCC<Nr>:FREQuency
driver.configure.lteMeas.rfSettings.scc.frequency.set(analyzer_freq = 1.0,␣
↪secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param analyzer_freq
No help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

6.1.1.9 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.configure.lteMeas.scc.repcap_secondaryCC_get()
driver.configure.lteMeas.scc.repcap_secondaryCC_set(repcap.SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.1.1.9.1 ChannelBw

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:SCC<Nr>:CBANdwidth
```

class ChannelBwCls

ChannelBw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*secondaryCC=SecondaryCC.Default*) → ChannelBandwidth

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SCC<Nr>:CBANdwidth
value: enums.ChannelBandwidth = driver.configure.lteMeas.scc.channelBw.
↪get(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
channel_bw: No help available

set(*channel_bw: ChannelBandwidth*, *secondaryCC=SecondaryCC.Default*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SCC<Nr>:CBANdwidth
driver.configure.lteMeas.scc.channelBw.set(channel_bw = enums.ChannelBandwidth.
↪B014, secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param channel_bw
No help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

6.1.1.10 Srs

SCPI Commands :

```
CONFigure:LTE:MEASurement<Instance>:SRS:VIEW
CONFigure:LTE:MEASurement<Instance>:SRS:TOUT
CONFigure:LTE:MEASurement<Instance>:SRS:REPetition
CONFigure:LTE:MEASurement<Instance>:SRS:SCONdition
CONFigure:LTE:MEASurement<Instance>:SRS:MOEXception
CONFigure:LTE:MEASurement<Instance>:SRS:HDMode
```

class SrsCls

Srs commands group definition. 8 total commands, 2 Subgroups, 6 group commands

get_hdmode() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:HDMode
value: bool = driver.configure.lteMeas.srs.get_hdmode()
```

Enables or disables the high dynamic mode for power dynamics measurements. With RF path sharing, this command is not applicable.

return
high_dynamic_mode: No help available

get_mo_exception() → bool

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:MOEXception
value: bool = driver.configure.lteMeas.srs.get_mo_exception()
```

Specifies whether measurement results that the CMX500 identifies as faulty or inaccurate are rejected.

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

get_repetition() → Repeat

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:REPetition
value: enums.Repeat = driver.configure.lteMeas.srs.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: Single-shot measurement CONTinuous: Continuous measurement

get_scondition() → StopCondition

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCONdition
value: enums.StopCondition = driver.configure.lteMeas.srs.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: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

get_timeout() → float

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:TOUT
value: float = driver.configure.lteMeas.srs.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. 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: No help available

get_view() → ViewSrs

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:VIEW
value: enums.ViewSrs = driver.configure.lteMeas.srs.get_view()
```

No command help available

return
view: No help available

set_hdmode(*high_dynamic_mode: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:HDMode
driver.configure.lteMeas.srs.set_hdmode(high_dynamic_mode = False)
```

Enables or disables the high dynamic mode for power dynamics measurements. With RF path sharing, this command is not applicable.

param high_dynamic_mode
No help available

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:MOEXception
driver.configure.lteMeas.srs.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the CMX500 identifies as faulty or inaccurate are rejected.

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

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:REPetition
driver.configure.lteMeas.srs.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: Single-shot measurement CONTinuous: Continuous measurement

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCONdition
driver.configure.lteMeas.srs.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: Continue measurement irrespective of the limit check. SLFail: Stop measurement on limit failure.

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:TOUT
driver.configure.lteMeas.srs.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. 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
No help available

set_view(*view: ViewSrs*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:VIEW
driver.configure.lteMeas.srs.set_view(view = enums.ViewSrs.PDYNamics)
```

No command help available

param view
No help available

Cloning the Group

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

Subgroups

6.1.1.10.1 Limit

class LimitCls

Limit commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.10.1.1 Pdynamics

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:SRS:LIMit:PDYNamics
```

class PdynamicsCls

Pdynamics commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class PdynamicsStruct

Response structure. Fields:

- Enable: bool: OFF: disables the limit check ON: enables the limit check
- On_Power_Upper: float: Upper limit for the ON power
- On_Power_Lower: float: Lower limit for the ON power
- Off_Power_Upper: float: Upper limit for the OFF power

get() → PdynamicsStruct

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:LIMit:PDYNamics
value: PdynamicsStruct = driver.configure.lteMeas.srs.limit.pdynamics.get()
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

return

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

set(*enable: bool*, *on_power_upper: float*, *on_power_lower: float*, *off_power_upper: float*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:LIMit:PDYNamics
driver.configure.lteMeas.srs.limit.pdynamics.set(enable = False, on_power_upper
↪= 1.0, on_power_lower = 1.0, off_power_upper = 1.0)
```

Defines limits for the ON power and OFF power determined with the power dynamics measurement.

param enable
OFF: disables the limit check ON: enables the limit check

param on_power_upper
Upper limit for the ON power

param on_power_lower
Lower limit for the ON power

param off_power_upper
Upper limit for the OFF power

6.1.1.10.2 Scount

SCPI Command :

```
CONFigure:LTE:MEASurement<Instance>:SRS:SCOunt:PDYNamics
```

class ScountCls

Scount commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_pdynamics() → int

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCOunt:PDYNamics
value: int = driver.configure.lteMeas.srs.scount.get_pdynamics()
```

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

return
statistic_count: Number of measurement intervals

set_pdynamics(*statistic_count: int*) → None

```
# SCPI: CONFigure:LTE:MEASurement<Instance>:SRS:SCOunt:PDYNamics
driver.configure.lteMeas.srs.scount.set_pdynamics(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
Number of measurement intervals

6.2 LteMeas

class LteMeasCls

LteMeas commands group definition. 795 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1 MultiEval

SCPI Commands :

```
INITiate:LTE:MEASurement<Instance>:MEValuation
STOP:LTE:MEASurement<Instance>:MEValuation
ABORt:LTE:MEASurement<Instance>:MEValuation
```

class MultiEvalCls

MultiEval commands group definition. 684 total commands, 19 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:LTE:MEASurement<Instance>:MEValuation
driver.lteMeas.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.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

initiate(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:LTE:MEASurement<Instance>:MEValuation
driver.lteMeas.multiEval.initiate()
```

(continues on next page)

(continued from previous page)

```
    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.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

stop() → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:MEValuation
driver.lteMeas.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.
    - 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(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:MEValuation
driver.lteMeas.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 RsCMPX_LteMeas.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

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

Subgroups

6.2.1.1 Aclr

class AclrCls

Aclr commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Utra_2_Neg: enums.ResultStatus2: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: enums.ResultStatus2: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: enums.ResultStatus2: ACLR for the first E-UTRA channel with lower frequency
- Eutra: enums.ResultStatus2: Power in the allocated E-UTRA channel
- Eutra_Positiv: enums.ResultStatus2: ACLR for the first E-UTRA channel with higher frequency
- Utra_1_Pos: enums.ResultStatus2: ACLR for the first UTRA channel with higher frequency

- Utra_2_Pos: enums.ResultStatus2: ACLR for the second UTRA channel with higher frequency

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Utra_2_Neg: float: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: float: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: float: ACLR for the first E-UTRA channel with lower frequency
- Eutra: float: Power in the allocated E-UTRA channel
- Eutra_Positiv: float: ACLR for the first E-UTRA channel with higher frequency
- Utra_1_Pos: float: ACLR for the first UTRA channel with higher frequency
- Utra_2_Pos: float: ACLR for the second UTRA channel with higher frequency

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.aclr.average.calculate()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: ResultData = driver.lteMeas.multiEval.aclr.average.fetch()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: ResultData = driver.lteMeas.multiEval.aclr.average.read()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'. 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.

6.2.1.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Utra_2_Neg: enums.ResultStatus2: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: enums.ResultStatus2: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: enums.ResultStatus2: ACLR for the first E-UTRA channel with lower frequency
- Eutra: enums.ResultStatus2: Power in the allocated E-UTRA channel
- Eutra_Positiv: enums.ResultStatus2: ACLR for the first E-UTRA channel with higher frequency
- Utra_1_Pos: enums.ResultStatus2: ACLR for the first UTRA channel with higher frequency
- Utra_2_Pos: enums.ResultStatus2: ACLR for the second UTRA channel with higher frequency

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Utra_2_Neg: float: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: float: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: float: ACLR for the first E-UTRA channel with lower frequency
- Eutra: float: Power in the allocated E-UTRA channel
- Eutra_Positiv: float: ACLR for the first E-UTRA channel with higher frequency
- Utra_1_Pos: float: ACLR for the first UTRA channel with higher frequency
- Utra_2_Pos: float: ACLR for the second UTRA channel with higher frequency

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.aclr.current.calculate()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: ResultData = driver.lteMeas.multiEval.aclr.current.fetch()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'. 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:LTE:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: ResultData = driver.lteMeas.multiEval.aclr.current.read()
```

Returns the relative ACLR values as displayed in the table below the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'. 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.

6.2.1.1.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Nr_Res_Blocks: int: Number of allocated resource blocks
- Offset_Res_Blocks: int: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.aclr.dallocation.fetch()
```

Returns the detected allocation for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

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

6.2.1.1.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → UplinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ACLR:DCHType
value: enums.UplinkChannelType = driver.lteMeas.multiEval.aclr.dchType.fetch()
```

Returns the uplink channel type for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.2 Amarker<AbsMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.amarker.repcap_absMarker_get()
driver.lteMeas.multiEval.amarker.repcap_absMarker_set(repcap.AbsMarker.Nr1)
```

class AmarkerCls

Amarker commands group definition. 6 total commands, 5 Subgroups, 0 group commands Repeated Capability: AbsMarker, default value after init: AbsMarker.Nr1

Cloning the Group

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

Subgroups

6.2.1.2.1 EvMagnitude

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude
```

class EvMagnitudeCls

EvMagnitude commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude
value: float = driver.lteMeas.multiEval.amarker.evMagnitude.fetch(xvalue = 1,␣
↪trace_select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.
↪Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Absolute y-value of the marker position

Cloning the Group

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

Subgroups

6.2.1.2.1.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:EVMagnitude:PEAK
value: float = driver.lteMeas.multiEval.amarker.evMagnitude.peak.fetch(xvalue =␣
↪1, trace_select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.
↪Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Absolute y-value of the marker position

6.2.1.2.2 Merror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:MERRor
```

class MerrorCls

Merror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:MERRor
value: float = driver.lteMeas.multiEval.amarker.merror.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Absolute y-value of the marker position

6.2.1.2.3 Pdynamics

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PDYNamics
```

class PdynamicsCls

Pdynamics commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: float*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PDYNamics
value: float = driver.lteMeas.multiEval.amarker.pdynamics.fetch(xvalue = 1.0,␣
↪trace_select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.
↪Default)
```

Uses the markers 1 and 2 with absolute values on the power dynamics trace.

Suppressed linked return values: reliability

param xvalue
(float or boolean) Absolute x-value of the marker position

param trace_select
No help available

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

return
yvalue: Absolute y-value of the marker position

6.2.1.2.4 Perror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PERRor
```

class PerrorCls

Perror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *absMarker=AbsMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PERRor
value: float = driver.lteMeas.multiEval.amarker.perror.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage, absMarker = repcap.AbsMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values

per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, . . . , symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Absolute y-value of the marker position

6.2.1.2.5 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.2.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.amarker.pmonitor.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.amarker.pmonitor.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*xvalue: int*, *absMarker=AbsMarker.Default*, *carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:AMARker<No>:PMONitor:CC<Nr>
value: float = driver.lteMeas.multiEval.amarker.pmonitor.cc.fetch(xvalue = 1,␣
↪absMarker = repcap.AbsMarker.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Uses the markers 1 and 2 with absolute values on the power monitor trace.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position (subframe number)

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

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

return
yvalue: Absolute y-value of the marker position

Cloning the Group

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

6.2.1.3 Bler

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:BLER
READ:LTE:MEASurement<Instance>:MEValuation:BLER
```

class BlerCls

Bler commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Ack: float: No parameter help available
- Nack: float: No parameter help available
- Bler: float: No parameter help available
- Dtx: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:BLER
value: ResultData = driver.lteMeas.multiEval.bler.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:BLER
value: ResultData = driver.lteMeas.multiEval.bler.read()
```

No command help available

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

6.2.1.4 Dmarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.dmarker.repcap_deltaMarker_get()
driver.lteMeas.multiEval.dmarker.repcap_deltaMarker_set(repcap.DeltaMarker.Nr1)
```

class DmarkerCls

Dmarker commands group definition. 6 total commands, 5 Subgroups, 0 group commands Repeated Capability: DeltaMarker, default value after init: DeltaMarker.Nr1

Cloning the Group

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

Subgroups

6.2.1.4.1 EvMagnitude

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude
```

class EvMagnitudeCls

EvMagnitude commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude
value: float = driver.lteMeas.multiEval.dmarker.evMagnitude.fetch(xvalue = 1,␣
↪trace_select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.
↪Default)
```

Uses the markers 1 and 2 with relative values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) X-value of the marker position relative to the x-value of the reference marker There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Y-value of the marker position relative to the y-value of the reference marker

Cloning the Group

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

Subgroups

6.2.1.4.1.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:EVMagnitude:PEAK
value: float = driver.lteMeas.multiEval.dmarker.evMagnitude.peak.fetch(xvalue =␣
↪1, trace_select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.
↪Default)
```

Uses the markers 1 and 2 with relative values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) X-value of the marker position relative to the x-value of the reference marker There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Y-value of the marker position relative to the y-value of the reference marker

6.2.1.4.2 Merror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:MERRor
```

class MerrorCls

Merror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:MERRor
value: float = driver.lteMeas.multiEval.dmarker.merror.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.Default)
```

Uses the markers 1 and 2 with relative values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) X-value of the marker position relative to the x-value of the reference marker There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Y-value of the marker position relative to the y-value of the reference marker

6.2.1.4.3 Pdynamics

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PDYNamics
```

class PdynamicsCls

Pdynamics commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: float*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PDYNamics
value: float = driver.lteMeas.multiEval.dmarker.pdynamics.fetch(xvalue = 1.0,␣
↪trace_select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.
↪Default)
```

Uses the markers 1 and 2 with relative values on the power dynamics trace.

Suppressed linked return values: reliability

param xvalue
(float or boolean) X-value of the marker position relative to the x-value of the reference marker

param trace_select
No help available

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

return
yvalue: Y-value of the marker position relative to the y-value of the reference marker

6.2.1.4.4 Perror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PERRor
```

class PerrorCls

Perror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PERRor
value: float = driver.lteMeas.multiEval.dmarker.perror.fetch(xvalue = 1, trace_
↪select = enums.TraceSelect.AVERage, deltaMarker = repcap.DeltaMarker.Default)
```

Uses the markers 1 and 2 with relative values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) X-value of the marker position relative to the x-value of the reference marker There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

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

return
yvalue: Y-value of the marker position relative to the y-value of the reference marker

6.2.1.4.5 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.4.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.dmarker.pmonitor.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.dmarker.pmonitor.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*xvalue: int*, *deltaMarker=DeltaMarker.Default*, *carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:DMARker<No>:PMONitor:CC<Nr>
value: float = driver.lteMeas.multiEval.dmarker.pmonitor.cc.fetch(xvalue = 1,␣
↪deltaMarker = repcap.DeltaMarker.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Uses the markers 1 and 2 with relative values on the power monitor trace.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) X-value of the marker position relative to the x-value of the reference marker (in subframes)

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

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

return
yvalue: Y-value of the marker position relative to the y-value of the reference marker

Cloning the Group

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

6.2.1.5 EsFlatness

class EsFlatnessCls

EsFlatness commands group definition. 13 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.5.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.esFlatness.average.calculate()
```

Return current, average and extreme single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
value: ResultData = driver.lteMeas.multiEval.esFlatness.average.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:AVERage
value: ResultData = driver.lteMeas.multiEval.esFlatness.average.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

6.2.1.5.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
```

class CurrentCls

Current commands group definition. 4 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.esFlatness.current.calculate()
```

Return current, average and extreme single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
value: ResultData = driver.lteMeas.multiEval.esFlatness.current.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent
value: ResultData = driver.lteMeas.multiEval.esFlatness.current.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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.lteMeas.multiEval.esFlatness.current.clone()
```

Subgroups

6.2.1.5.2.1 ScIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent:SCINdex
```

class ScIndexCls

ScIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Maximum_1: int: SC index of Max (Range 1)
- Minimum_1: int: SC index of Min (Range 1)
- Maximum_2: int: SC index of Max (Range 2)
- Minimum_2: int: SC index of Min (Range 2)

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:CURRent:SCINdex
value: FetchStruct = driver.lteMeas.multiEval.esFlatness.current.scIndex.fetch()
```

Returns subcarrier indices of the equalizer spectrum flatness measurement. At these SC indices, the current minimum and maximum power of the equalizer coefficients have been detected within range 1 and range 2.

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

6.2.1.5.3 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
```

class ExtremeCls

Extreme commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
value: CalculateStruct = driver.lteMeas.multiEval.esFlatness.extreme.calculate()
```

Return current, average and extreme single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
value: ResultData = driver.lteMeas.multiEval.esFlatness.extreme.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:EXTReme
value: ResultData = driver.lteMeas.multiEval.esFlatness.extreme.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

6.2.1.5.4 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Ripple_1: float or bool: No parameter help available
- Ripple_2: float or bool: No parameter help available
- Max_R_1_Min_R_2: float or bool: No parameter help available
- Max_R_2_Min_R_1: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.esFlatness.standardDev.
↪calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
value: ResultData = driver.lteMeas.multiEval.esFlatness.standardDev.fetch()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:ESFLatness:SDEViation
value: ResultData = driver.lteMeas.multiEval.esFlatness.standardDev.read()
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement. See also 'Equalizer spectrum flatness limits'.

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

6.2.1.6 EvMagnitude

class EvMagnitudeCls

EvMagnitude commands group definition. 21 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.6.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
```

class AverageCls

Average commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position
- High: List[float]: EVM value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.evMagnitude.average.
↪calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
value: ResultData = driver.lteMeas.multiEval.evMagnitude.average.fetch()
```

Returns the values of the EVM RMS diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage
value: ResultData = driver.lteMeas.multiEval.evMagnitude.average.read()
```

Returns the values of the EVM RMS diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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.lteMeas.multiEval.evMagnitude.average.clone()
```

Subgroups

6.2.1.6.1.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available

- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage:NREF
value: ResultData = driver.lteMeas.multiEval.evMagnitude.average.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:AVERage:NREF
value: ResultData = driver.lteMeas.multiEval.evMagnitude.average.nref.read()
```

No command help available

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

6.2.1.6.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
```

class CurrentCls

Current commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position

- High: List[float]: EVM value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.evMagnitude.current.
↪calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
value: ResultData = driver.lteMeas.multiEval.evMagnitude.current.fetch()
```

Returns the values of the EVM RMS diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent
value: ResultData = driver.lteMeas.multiEval.evMagnitude.current.read()
```

Returns the values of the EVM RMS diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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.lteMeas.multiEval.evMagnitude.current.clone()
```

Subgroups

6.2.1.6.2.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent:NREF
value: ResultData = driver.lteMeas.multiEval.evMagnitude.current.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:CURRent:NREF
value: ResultData = driver.lteMeas.multiEval.evMagnitude.current.nref.read()
```

No command help available

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

6.2.1.6.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
```

class MaximumCls

Maximum commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position
- High: List[float]: EVM value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.evMagnitude.maximum.
↪calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
value: ResultData = driver.lteMeas.multiEval.evMagnitude.maximum.fetch()
```

Returns the values of the EVM RMS diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum
value: ResultData = driver.lteMeas.multiEval.evMagnitude.maximum.read()
```

Returns the values of the EVM RMS diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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.lteMeas.multiEval.evMagnitude.maximum.clone()
```

Subgroups

6.2.1.6.3.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum:NREF
value: ResultData = driver.lteMeas.multiEval.evMagnitude.maximum.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:MAXimum:NREF
value: ResultData = driver.lteMeas.multiEval.evMagnitude.maximum.nref.read()
```

No command help available

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

6.2.1.6.4 Peak

class PeakCls

Peak commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.6.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position
- High: List[float]: EVM value for high EVM window position

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
value: ResultData = driver.lteMeas.multiEval.evMagnitude.peak.average.fetch()
```

Returns the values of the EVM peak diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'Square EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:AVERage
value: ResultData = driver.lteMeas.multiEval.evMagnitude.peak.average.read()
```

Returns the values of the EVM peak diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

6.2.1.6.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position
- High: List[float]: EVM value for high EVM window position

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
value: ResultData = driver.lteMeas.multiEval.evMagnitude.peak.current.fetch()
```

Returns the values of the EVM peak diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:CURRent
value: ResultData = driver.lteMeas.multiEval.evMagnitude.peak.current.read()
```

Returns the values of the EVM peak diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

6.2.1.6.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position
- High: List[float]: EVM value for high EVM window position

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
value: ResultData = driver.lteMeas.multiEval.evMagnitude.peak.maximum.fetch()
```

Returns the values of the EVM peak diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMagnitude:PEAK:MAXimum
value: ResultData = driver.lteMeas.multiEval.evMagnitude.peak.maximum.read()
```

Returns the values of the EVM peak diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square EVM'.

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

6.2.1.7 Evmc

class EvmcCls

Evmc commands group definition. 8 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.7.1 Peak

class PeakCls

Peak commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.7.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
value: float = driver.lteMeas.multiEval.evmc.peak.average.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_average: No help available

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:AVERage
value: float = driver.lteMeas.multiEval.evmc.peak.average.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_average: No help available

6.2.1.7.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
value: float = driver.lteMeas.multiEval.evmc.peak.current.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_current: No help available

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:CURRent
value: float = driver.lteMeas.multiEval.evmc.peak.current.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_current: No help available

6.2.1.7.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
value: float = driver.lteMeas.multiEval.evmc.peak.maximum.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_maximum: No help available

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:MAXimum
value: float = driver.lteMeas.multiEval.evmc.peak.maximum.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_maximum: No help available

6.2.1.7.1.4 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
value: float = driver.lteMeas.multiEval.evmc.peak.standardDev.fetch()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_std_dev: No help available

read() → float

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:EVMC:PEAK:SDEViation
value: float = driver.lteMeas.multiEval.evmc.peak.standardDev.read()
```

The CURRent command returns the maximum value of the EVM vs subcarrier trace. The AVERage, MAXimum and SDEViation values are calculated from the CURRent values. The peak results cannot be displayed at the GUI.

Suppressed linked return values: reliability

return
evm_cpeak_std_dev: No help available

6.2.1.8 InbandEmission

class InbandEmissionCls

InbandEmission commands group definition. 30 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.inbandEmission.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.inbandEmission.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.2.1.8.1.1 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.1.2 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Margin: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.cc.margin.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the in-band emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

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

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

6.2.1.8.1.3 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Margin: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.cc.margin.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the in-band emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

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

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.lteMeas.multiEval.inbandEmission.cc.margin.current.clone()
```

Subgroups

6.2.1.8.1.4 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Rb_Index: int: Resource block index

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.cc.margin.current.
↪rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices for CC<no> in-band emission margins. At these RB indices, the CURRent and EXTReme margins have been detected (see method RsCMPX_LteMeas.LteMeas.MultiEval.InbandEmission.Cc.Margin.Current.fetch and ...:EXTReme) .

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

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

6.2.1.8.1.5 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Margin: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.cc.margin.extreme.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the in-band emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

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

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.lteMeas.multiEval.inbandEmission.cc.margin.extreme.clone()
```

Subgroups

6.2.1.8.1.6 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Rb_Index: int: Resource block index

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.cc.margin.extreme.
↪rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices for CC<no> in-band emission margins. At these RB indices, the CURRent and EXTReme margins have been detected (see method RsCMPX_LteMeas.LteMeas.MultiEval.InbandEmission.Cc.Margin.Current.fetch and . . . :EXTReme) .

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

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

6.2.1.8.1.7 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Margin: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:CC<Nr>
↪:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.cc.margin.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the limit line margin results for the CC<no> diagram. The CURRent margin indicates the minimum (vertical) distance between the in-band emissions limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins. The margin results cannot be displayed at the GUI.

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

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

6.2.1.8.2 Pcc

class PccCls

Pcc commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.2.1 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.2.2 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission[:PCC]:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission[:PCC]:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.pcc.margin.average.
↪fetch()
```

No command help available

return

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

6.2.1.8.2.3 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission[:PCC]:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission[:PCC]:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.pcc.margin.current.
↪fetch()
```

No command help available

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.lteMeas.multiEval.inbandEmission.pcc.margin.current.clone()
```

Subgroups

6.2.1.8.2.4 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission[:PCC]:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission[:PCC]:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.pcc.margin.current.
↪rbIndex.fetch()
```

No command help available

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

6.2.1.8.2.5 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission[:PCC]:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission[:PCC]:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.pcc.margin.extreme.
↪fetch()
```

No command help available

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.lteMeas.multiEval.inbandEmission.pcc.margin.extreme.clone()
```

Subgroups

6.2.1.8.2.6 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission[:PCC]:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission[:PCC]:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.pcc.margin.extreme.
↪rbIndex.fetch()
```

No command help available

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

6.2.1.8.2.7 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission[:PCC]:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission[:PCC]:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.pcc.margin.
↪standardDev.fetch()
```

No command help available

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

6.2.1.8.3 Scc

class SccCls

Scc commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.3.1 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.3.2 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.scc.margin.average.
↪fetch()
```

No command help available

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

6.2.1.8.3.3 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.scc.margin.current.
↪fetch()
```

No command help available

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.lteMeas.multiEval.inbandEmission.scc.margin.current.clone()
```

Subgroups

6.2.1.8.3.4 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:SCC:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.scc.margin.current.
↪rbIndex.fetch()
```

No command help available

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

6.2.1.8.3.5 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.scc.margin.extreme.
↪fetch()
```

No command help available

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.lteMeas.multiEval.inbandEmission.scc.margin.extreme.clone()
```

Subgroups

6.2.1.8.3.6 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:SCC:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.scc.margin.extreme.
↪rbIndex.fetch()
```

No command help available

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

6.2.1.8.3.7 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:SCC:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:SCC:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.scc.margin.
↪standardDev.fetch()
```

No command help available

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

6.2.1.8.4 Ulca

class UlcaCls

Ulca commands group definition. 12 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.4.1 Pcc

class PccCls

Pcc commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.8.4.2 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.clone()
```

Subgroups

6.2.1.8.4.3 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA[:PCC]:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:ULCA[:PCC]:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.
↪average.fetch()
```

No command help available

return

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

6.2.1.8.4.4 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA[:PCC]:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:ULCA[:PCC]:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.
↪current.fetch()
```

No command help available

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.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.current.clone()
```

Subgroups

6.2.1.8.4.5 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA[:PCC]:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:ULCA[:PCC]:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.
↪current.rbIndex.fetch()
```

No command help available

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

6.2.1.8.4.6 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA[:PCC]:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:ULCA[:PCC]:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.
↪extreme.fetch()
```

No command help available

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.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.extreme.clone()
```

Subgroups

6.2.1.8.4.7 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA[:PCC]:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:ULCA[:PCC]:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.
↪extreme.rbIndex.fetch()
```

No command help available

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

6.2.1.8.4.8 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA[:PCC]:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:IEMission:ULCA[:PCC]:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.pcc.margin.
↪standardDev.fetch()
```

No command help available

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

6.2.1.8.4.9 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.lteMeas.multiEval.inbandEmission.ulca.scc.repcap_secondaryCC_get()
driver.lteMeas.multiEval.inbandEmission.ulca.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.2.1.8.4.10 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.inbandEmission.ulca.scc.margin.clone()
```

Subgroups

6.2.1.8.4.11 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>
↪:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.scc.margin.
↪average.fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.8.4.12 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>
↪:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.scc.margin.
↪current.fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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.lteMeas.multiEval.inbandEmission.ulca.scc.margin.current.clone()
```

Subgroups

6.2.1.8.4.13 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>
↪:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.scc.margin.
↪current.rbIndex.fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.8.4.14 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available

- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>
↪:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.scc.margin.
↪extreme.fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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.lteMeas.multiEval.inbandEmission.ulca.scc.margin.extreme.clone()
```

Subgroups

6.2.1.8.4.15 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>
↪:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.scc.margin.
↪extreme.rbIndex.fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.8.4.16 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:IEMission:ULCA:SCC<Nr>
↪:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.inbandEmission.ulca.scc.margin.
↪standardDev.fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.9 ListPy

class ListPyCls

ListPy commands group definition. 364 total commands, 9 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.1 Aclr

class AclrCls

Aclr commands group definition. 22 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.1.1 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Nr_Res_Blocks: List[int]: Number of allocated resource blocks
- Offset_Res_Blocks: List[int]: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.listPy.aclr.dallocation.fetch()
```

Return the detected allocation for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ. The results are returned as pairs per segment: <Reliability>, {<NrResBlocks>, <OffsetResBlocks>}Seg 1, {<NrResBlocks>, <OffsetResBlocks>}Seg 2, ...

return

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

6.2.1.9.1.2 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[UplinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:DCHType
value: List[enums.UplinkChannelType] = driver.lteMeas.multiEval.listPy.aclr.
↪dchType.fetch()
```

Return the uplink channel type for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

Suppressed linked return values: reliability

return
channel_type: Comma-separated list of values, one per measured segment

6.2.1.9.1.3 Eutra

class EutraCls

Eutra commands group definition. 12 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.1.4 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.eutra.
↪average.calculate()
```

Return the power in the allocated E-UTRA channel 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.

Suppressed linked return values: reliability

return
eutra: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.eutra.average.fetch()
```

Return the power in the allocated E-UTRA channel 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.

Suppressed linked return values: reliability

return
eutra: Comma-separated list of values, one per measured segment

6.2.1.9.1.5 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.eutra.
↪current.calculate()
```

Return the power in the allocated E-UTRA channel 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.

Suppressed linked return values: reliability

return
eutra: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.eutra.current.fetch()
```

Return the power in the allocated E-UTRA channel 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.

Suppressed linked return values: reliability

return
eutra: Comma-separated list of values, one per measured segment

6.2.1.9.1.6 Negativ

class NegativCls

Negativ commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.1.7 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.eutra.
↪negativ.average.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_negativ: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.eutra.negativ.average.
↪fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_negativ: Comma-separated list of values, one per measured segment

6.2.1.9.1.8 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.eutra.
↪negativ.current.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_negativ: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:NEGativ:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.eutra.negativ.current.
↪fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_negativ: Comma-separated list of values, one per measured segment

6.2.1.9.1.9 Positiv

class PositivCls

Positiv commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.1.10 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.eutra.
↪positiv.average.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_positiv: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.eutra.positiv.average.
↪fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_positiv: Comma-separated list of values, one per measured segment

6.2.1.9.1.11 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.eutra.
↪positiv.current.calculate()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_positiv: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:EUTRa:POSitiv:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.eutra.positiv.current.
↪fetch()
```

Return the ACLR for the first adjacent E-UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

return
eutra_positiv: Comma-separated list of values, one per measured segment

6.2.1.9.1.12 Utra<UtraAdjChannel>

RepCap Settings

```
# Range: Ch1 .. Ch2
rc = driver.lteMeas.multiEval.listPy.aclr.utra.repcap_utraAdjChannel_get()
driver.lteMeas.multiEval.listPy.aclr.utra.repcap_utraAdjChannel_set(repcap.
↪UtraAdjChannel.Ch1)
```

class UtraCls

Utra commands group definition. 8 total commands, 2 Subgroups, 0 group commands Repeated Capability: UtraAdjChannel, default value after init: UtraAdjChannel.Ch1

Cloning the Group

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

Subgroups

6.2.1.9.1.13 Negativ

class NegativCls

Negativ commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.1.14 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:AVERage
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.utra.
↪negativ.average.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.utra.negativ.average.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

6.2.1.9.1.15 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:NEGativ:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:CURRent
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.utra.
↪negativ.current.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:NEGativ:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.utra.negativ.current.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

6.2.1.9.1.16 Positiv

class PositivCls

Positiv commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.1.17 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:AVERage
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.utra.
↪positiv.average.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.utra.positiv.average.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

6.2.1.9.1.18 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>:POSitiv:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*utraAdjChannel=UtraAdjChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:CURRent
value: List[enums.ResultStatus2] = driver.lteMeas.multiEval.listPy.aclr.utra.
↪positiv.current.calculate(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

fetch(*utraAdjChannel=UtraAdjChannel.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr>
↪:POSitiv:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.aclr.utra.positiv.current.
↪fetch(utraAdjChannel = repcap.UtraAdjChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency 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.

Suppressed linked return values: reliability

param utraAdjChannel
optional repeated capability selector. Default value: Ch1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

6.2.1.9.2 EsFlatness

class EsFlatnessCls

EsFlatness commands group definition. 30 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.2.1 Difference<Difference>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.listPy.esFlatness.difference.repcap_difference_get()
driver.lteMeas.multiEval.listPy.esFlatness.difference.repcap_difference_set(repcap.
↪Difference.Nr1)
```

class DifferenceCls

Difference commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: Difference, default value after init: Difference.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.2.2 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:DIFFerence<nr>:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.
↪difference.average.calculate(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) 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.

Suppressed linked return values: reliability

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

return
difference: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.difference.
↪average.fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) 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.

Suppressed linked return values: reliability

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

return
difference: Comma-separated list of values, one per measured segment

6.2.1.9.2.3 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

`calculate`(*difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:DIFFerence<nr>:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.
↪difference.current.calculate(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) 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.

Suppressed linked return values: reliability

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

return
difference: (float or boolean items) Comma-separated list of values, one per measured segment

`fetch`(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.difference.
↪current.fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) 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.

Suppressed linked return values: reliability

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

return
difference: Comma-separated list of values, one per measured segment

6.2.1.9.2.4 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:DIFFerence<nr>:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.
↪difference.extreme.calculate(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) 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.

Suppressed linked return values: reliability

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

return
difference: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.difference.
↪extreme.fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) 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.

Suppressed linked return values: reliability

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

return
difference: Comma-separated list of values, one per measured segment

6.2.1.9.2.5 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence<nr>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:DIFFerence
↪<nr>:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.difference.
↪standardDev.fetch(difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) 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.

Suppressed linked return values: reliability

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

return
difference: Comma-separated list of values, one per measured segment

6.2.1.9.2.6 Maxr<MaxRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.listPy.esFlatness.maxr.repcap_maxRange_get()
driver.lteMeas.multiEval.listPy.esFlatness.maxr.repcap_maxRange_set(repcap.MaxRange.Nr1)
```

class MaxrCls

Maxr commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: MaxRange, default value after init: MaxRange.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.2.7 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.maxr.
↪average.calculate(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) 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.

Suppressed linked return values: reliability

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

return
maxr: (float or boolean items) Comma-separated list of values, one per measured segment.

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.maxr.average.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) 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.

Suppressed linked return values: reliability

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

return
maxr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.8 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.maxr.
↪current.calculate(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) 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.

Suppressed linked return values: reliability

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

return
maxr: (float or boolean items) Comma-separated list of values, one per measured segment.

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.maxr.current.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) 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.

Suppressed linked return values: reliability

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

return
maxr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.9 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.maxr.
↪extreme.calculate(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) 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.

Suppressed linked return values: reliability

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

return
maxr: (float or boolean items) Comma-separated list of values, one per measured segment.

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.maxr.extreme.
↪fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) 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.

Suppressed linked return values: reliability

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

return
maxr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.10 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MAXR<nr>
↪:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.maxr.
↪standardDev.fetch(maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) 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.

Suppressed linked return values: reliability

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

return
maxr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.11 Minr<MinRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.listPy.esFlatness.minr.repcap_minRange_get()
driver.lteMeas.multiEval.listPy.esFlatness.minr.repcap_minRange_set(repcap.MinRange.Nr1)
```

class MinrCls

Minr commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: MinRange, default value after init: MinRange.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.2.12 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.minr.
↪average.calculate(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) 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.

Suppressed linked return values: reliability

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

return
minr: (float or boolean items) Comma-separated list of values, one per measured segment.

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.minr.average.
↪fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) 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.

Suppressed linked return values: reliability

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

return
minr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.13 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.minr.
↪current.calculate(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) 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.

Suppressed linked return values: reliability

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

return
minr: (float or boolean items) Comma-separated list of values, one per measured segment.

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.minr.current.
↪fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) 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.

Suppressed linked return values: reliability

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

return
minr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.14 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.minr.
↪extreme.calculate(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) 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.

Suppressed linked return values: reliability

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

return
minr: (float or boolean items) Comma-separated list of values, one per measured segment.

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.minr.extreme.
↪fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) 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.

Suppressed linked return values: reliability

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

return
minr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.15 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:MINR<nr>
↪:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.minr.
↪standardDev.fetch(minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) 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.

Suppressed linked return values: reliability

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

return
minr: Comma-separated list of values, one per measured segment.

6.2.1.9.2.16 Ripple<Ripple>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.listPy.esFlatness.ripple.repcap_ripple_get()
driver.lteMeas.multiEval.listPy.esFlatness.ripple.repcap_ripple_set(repcap.Ripple.Nr1)
```

class RippleCls

Ripple commands group definition. 7 total commands, 4 Subgroups, 0 group commands Repeated Capability: Ripple, default value after init: Ripple.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.2.17 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle
↪<nr>:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.ripple.
↪average.calculate(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) 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.

Suppressed linked return values: reliability

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

return
ripple: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.ripple.average.
↪fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) 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.

Suppressed linked return values: reliability

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

return
ripple: Comma-separated list of values, one per measured segment

6.2.1.9.2.18 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle
↪<nr>:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.ripple.
↪current.calculate(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) 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.

Suppressed linked return values: reliability

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

return
ripple: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.ripple.current.
↪fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) 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.

Suppressed linked return values: reliability

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

return
ripple: Comma-separated list of values, one per measured segment

6.2.1.9.2.19 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle
↪<nr>:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.esFlatness.ripple.
↪extreme.calculate(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) 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.

Suppressed linked return values: reliability

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

return
ripple: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.ripple.extreme.
↪fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) 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.

Suppressed linked return values: reliability

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

return
ripple: Comma-separated list of values, one per measured segment

6.2.1.9.2.20 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:RIPPle<nr>
↪:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.esFlatness.ripple.
↪standardDev.fetch(ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) 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.

Suppressed linked return values: reliability

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

return
ripple: Comma-separated list of values, one per measured segment

6.2.1.9.2.21 ScIndex

class ScIndexCls

ScIndex commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.2.22 Maximum<MaxRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.listPy.esFlatness.scIndex.maximum.repcap_maxRange_get()
driver.lteMeas.multiEval.listPy.esFlatness.scIndex.maximum.repcap_maxRange_set(repcap.
↪MaxRange.Nr1)
```

class MaximumCls

Maximum commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: MaxRange, default value after init: MaxRange.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.2.23 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:SCINdex:MAXimum<nr>:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*maxRange=MaxRange.Default*) → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:SCINdex:MAXimum<nr>:CURRent
value: List[int] = driver.lteMeas.multiEval.listPy.esFlatness.scIndex.maximum.
↪current.fetch(maxRange = repcap.MaxRange.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for all measured list mode segments. At these SC indices, the current MINimum or MAXimum power of the equalizer coefficients has been detected within the selected range.

Suppressed linked return values: reliability

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

return
maximum: Comma-separated list of values, one per measured segment

6.2.1.9.2.24 Minimum<MinRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.lteMeas.multiEval.listPy.esFlatness.scIndex.minimum.repcap_minRange_get()
driver.lteMeas.multiEval.listPy.esFlatness.scIndex.minimum.repcap_minRange_set(repcap.
↪MinRange.Nr1)
```

class MinimumCls

Minimum commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: MinRange, default value after init: MinRange.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.2.25 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:ESFLatness:SCINdex:MINimum<nr>:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*minRange=MinRange.Default*) → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:ESFLatness:SCINdex:MINimum<nr>:CURRent
value: List[int] = driver.lteMeas.multiEval.listPy.esFlatness.scIndex.minimum.
↪current.fetch(minRange = repcap.MinRange.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for all measured list mode segments. At these SC indices, the current MINimum or MAXimum power of the equalizer coefficients has been detected within the selected range.

Suppressed linked return values: reliability

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

return
minimum: Comma-separated list of values, one per measured segment

6.2.1.9.3 InbandEmission

class InbandEmissionCls

InbandEmission commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.3.1 Margin

class MarginCls

Margin commands group definition. 6 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.3.2 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.inbandEmission.margin.
↪average.fetch()
```

Return the in-band emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Suppressed linked return values: reliability

return
margin: Comma-separated list of values, one per measured segment

6.2.1.9.3.3 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.inbandEmission.margin.
↪current.fetch()
```

Return the in-band emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Suppressed linked return values: reliability

return
margin: Comma-separated list of values, one per measured segment

6.2.1.9.3.4 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.inbandEmission.margin.
↪extreme.fetch()
```

Return the in-band emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Suppressed linked return values: reliability

return
margin: Comma-separated list of values, one per measured segment

6.2.1.9.3.5 RbIndex

class RbIndexCls

RbIndex commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.3.6 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:RBINdex:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:RBINdex:CURRent
value: List[int] = driver.lteMeas.multiEval.listPy.inbandEmission.margin.
↪rbIndex.current.fetch()
```

Return resource block indices of the in-band emission measurement for all measured list mode segments. At these RB indices, the CURRent and EXTReme margins have been detected.

Suppressed linked return values: reliability

return
rb_index: Comma-separated list of values, one per measured segment

6.2.1.9.3.7 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:RBINdex:EXTReme
```

class ExtremeCls

Extreme commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:RBINdex:EXTReme
value: List[int] = driver.lteMeas.multiEval.listPy.inbandEmission.margin.
↪rbIndex.extreme.fetch()
```

Return resource block indices of the in-band emission measurement for all measured list mode segments. At these RB indices, the CURRent and EXTReme margins have been detected.

Suppressed linked return values: reliability

return
rb_index: Comma-separated list of values, one per measured segment

6.2.1.9.3.8 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:IEMission:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:IEMission:MARGin:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.inbandEmission.margin.
↪standardDev.fetch()
```

Return the in-band emission limit line margin results for all measured list mode segments. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

Suppressed linked return values: reliability

return
margin: Comma-separated list of values, one per measured segment

6.2.1.9.4 Modulation

class ModulationCls

Modulation commands group definition. 178 total commands, 13 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.1 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Nr_Res_Blocks: List[int]: Number of allocated resource blocks
- Offset_Res_Blocks: List[int]: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.listPy.modulation.dallocation.
↪fetch()
```

Return the detected allocation for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ. The results are returned as pairs per segment: <Reliability>, {<NrResBlocks>, <OffsetResBlocks>}Seg 1, {<NrResBlocks>, <OffsetResBlocks>}Seg 2, ...

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

6.2.1.9.4.2 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[UplinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DCHType
value: List[enums.UplinkChannelType] = driver.lteMeas.multiEval.listPy.
↪modulation.dchType.fetch()
```

Return the uplink channel type for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result

when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

Suppressed linked return values: reliability

return
channel_type: Comma-separated list of values, one per measured segment

6.2.1.9.4.3 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DMODulation
```

class DmodulationCls

Dmodulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[Modulation]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:DMODulation
value: List[enums.Modulation] = driver.lteMeas.multiEval.listPy.modulation.
↪dmodulation.fetch()
```

Return the detected modulation scheme for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. If channel type PUCCH is detected, QPSK is returned for the modulation scheme because the QPSK limits are applied in that case.

Suppressed linked return values: reliability

return
modulation: Comma-separated list of values, one per measured segment QPSK, 16QAM, 64QAM, 256QAM

6.2.1.9.4.4 Evm

class EvmCls

Evm commands group definition. 42 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.5 Dmrs

class DmrsCls

Dmrs commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.6 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.high.clone()
```

Subgroups

6.2.1.9.4.7 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪dmrs.high.average.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.high.
↪average.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.8 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪dmrs.high.current.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.high.
↪current.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.9 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪dmrs.high.extreme.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.high.
↪extreme.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.10 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.high.
↪standardDev.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.11 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.low.clone()
```

Subgroups

6.2.1.9.4.12 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪dmrs.low.average.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.low.
↪average.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.13 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪dmrs.low.current.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.low.
↪current.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.14 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪dmrs.low.extreme.calculate()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.low.
↪extreme.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.15 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:DMRS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:DMRS:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.dmrs.low.
↪standardDev.fetch()
```

Return error vector magnitude DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.16 Peak

class PeakCls

Peak commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.17 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.evm.peak.high.clone()
```

Subgroups

6.2.1.9.4.18 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪peak.high.average.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.high.
↪average.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.19 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪peak.high.current.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.high.
↪current.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.20 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪peak.high.extreme.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.high.
↪extreme.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.21 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.high.
↪standardDev.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.22 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.evm.peak.low.clone()
```

Subgroups

6.2.1.9.4.23 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪peak.low.average.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.low.
↪average.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.24 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪peak.low.current.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.low.
↪current.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.25 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.
↪peak.low.extreme.calculate()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.low.
↪extreme.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.26 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:PEAK:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:PEAK:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.peak.low.
↪standardDev.fetch()
```

Return error vector magnitude peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.27 Rms

class RmsCls

Rms commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.28 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.evm.rms.high.clone()
```

Subgroups

6.2.1.9.4.29 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.
↪high.average.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.high.
↪average.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.30 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.
↪high.current.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.high.
↪current.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.31 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.
↪high.extreme.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.high.
↪extreme.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.32 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.high.
↪standardDev.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.33 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.evm.rms.low.clone()
```

Subgroups

6.2.1.9.4.34 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.
↪low.average.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.low.
↪average.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.35 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.
↪low.current.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.low.
↪current.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.36 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.
↪low.extreme.calculate()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.low.
↪extreme.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.37 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EVM:RMS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:EVM:RMS:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.evm.rms.low.
↪standardDev.fetch()
```

Return error vector magnitude RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.38 FreqError

class FreqErrorCls

FreqError commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.39 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.
↪freqError.average.calculate()
```

Return carrier frequency 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.

Suppressed linked return values: reliability

return
frequency_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.freqError.
↪average.fetch()
```

Return carrier frequency 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.

Suppressed linked return values: reliability

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.40 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.
↪freqError.current.calculate()
```

Return carrier frequency 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.

Suppressed linked return values: reliability

return
frequency_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.freqError.
↪current.fetch()
```

Return carrier frequency 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.

Suppressed linked return values: reliability

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.41 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.
↪freqError.extreme.calculate()
```

Return carrier frequency 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.

Suppressed linked return values: reliability

return
frequency_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.freqError.
↪extreme.fetch()
```

Return carrier frequency 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.

Suppressed linked return values: reliability

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.42 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:FERRor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:FERRor:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.freqError.
↪standardDev.fetch()
```

Return carrier frequency 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.

Suppressed linked return values: reliability

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.43 IqOffset

class IqOffsetCls

IqOffset commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.44 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.
↪iqOffset.average.calculate()
```

Return I/Q origin offset 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.

Suppressed linked return values: reliability

return
iq_offset: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.iqOffset.
↪average.fetch()
```

Return I/Q origin offset 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.

Suppressed linked return values: reliability

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.9.4.45 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.
↪iqOffset.current.calculate()
```

Return I/Q origin offset 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.

Suppressed linked return values: reliability

return
iq_offset: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.iqOffset.
↪current.fetch()
```

Return I/Q origin offset 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.

Suppressed linked return values: reliability

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.9.4.46 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.
↪iqOffset.extreme.calculate()
```

Return I/Q origin offset 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.

Suppressed linked return values: reliability

return
iq_offset: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.iqOffset.
↪extreme.fetch()
```

Return I/Q origin offset 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.

Suppressed linked return values: reliability

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.9.4.47 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:IQOFfset:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:IQOFfset:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.iqOffset.
↪standardDev.fetch()
```

Return I/Q origin offset 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.

Suppressed linked return values: reliability

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.9.4.48 Merror

class MerrorCls

Merror commands group definition. 42 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.49 Dmrs

class DmrsCls

Dmrs commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.50 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.high.clone()
```

Subgroups

6.2.1.9.4.51 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪dmrs.high.average.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.
↪high.average.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.52 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪dmrs.high.current.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.
↪high.current.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.53 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪dmrs.high.extreme.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.
↪high.extreme.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.54 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.
↪high.standardDev.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.55 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.low.clone()
```

Subgroups

6.2.1.9.4.56 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪dmrs.low.average.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.low.
↪average.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.57 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪dmrs.low.current.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.low.
↪current.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.58 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪dmrs.low.extreme.calculate()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.low.
↪extreme.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.59 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:DMRS:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.dmrs.low.
↪standardDev.fetch()
```

Return magnitude error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.60 Peak

class PeakCls

Peak commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.61 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.merror.peak.high.clone()
```

Subgroups

6.2.1.9.4.62 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪peak.high.average.calculate()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.
↪high.average.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.63 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪peak.high.current.calculate()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.
↪high.current.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.64 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪peak.high.extreme.calculate()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.
↪high.extreme.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.65 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.
↪high.standardDev.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.66 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.merror.peak.low.clone()
```

Subgroups

6.2.1.9.4.67 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪peak.low.average.calculate()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.low.
↪average.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.68 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪peak.low.current.calculate()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.low.
↪current.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.69 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪peak.low.extreme.calculate()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.low.
↪extreme.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.70 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:PEAK:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.peak.low.
↪standardDev.fetch()
```

Return magnitude error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.71 Rms

class RmsCls

Rms commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.72 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.merror.rms.high.clone()
```

Subgroups

6.2.1.9.4.73 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪rms.high.average.calculate()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.high.
↪average.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.74 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪rms.high.current.calculate()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.high.
↪current.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.75 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪rms.high.extreme.calculate()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.high.
↪extreme.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.76 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.high.
↪standardDev.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.77 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.merror.rms.low.clone()
```

Subgroups

6.2.1.9.4.78 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪rms.low.average.calculate()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.low.
↪average.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.79 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪rms.low.current.calculate()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.low.
↪current.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.80 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.merror.
↪rms.low.extreme.calculate()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.low.
↪extreme.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.81 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:MERRor:RMS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:MERRor:RMS:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.merror.rms.low.
↪standardDev.fetch()
```

Return magnitude error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.82 Perror

class PerrorCls

Perror commands group definition. 42 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.83 Dmrs

class DmrsCls

Dmrs commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.84 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.high.clone()
```

Subgroups

6.2.1.9.4.85 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪dmrs.high.average.calculate()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.
↪high.average.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.86 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪dmrs.high.current.calculate()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.
↪high.current.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.87 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪dmrs.high.extreme.calculate()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.
↪high.extreme.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.88 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.
↪high.standardDev.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.89 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.low.clone()
```

Subgroups

6.2.1.9.4.90 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪dmrs.low.average.calculate()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.low.
↪average.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.91 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪dmrs.low.current.calculate()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.low.
↪current.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.92 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪dmrs.low.extreme.calculate()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.low.
↪extreme.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.93 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:DMRS:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.dmrs.low.
↪standardDev.fetch()
```

Return phase error DMRS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.94 Peak

class PeakCls

Peak commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.95 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.perror.peak.high.clone()
```

Subgroups

6.2.1.9.4.96 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪peak.high.average.calculate()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.
↪high.average.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.97 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪peak.high.current.calculate()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.
↪high.current.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.98 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪peak.high.extreme.calculate()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.
↪high.extreme.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.99 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.
↪high.standardDev.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.100 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.perror.peak.low.clone()
```

Subgroups

6.2.1.9.4.101 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪peak.low.average.calculate()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.low.
↪average.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.102 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪peak.low.current.calculate()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.low.
↪current.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.103 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪peak.low.extreme.calculate()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.low.
↪extreme.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.104 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:PEAK:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.peak.low.
↪standardDev.fetch()
```

Return phase error peak values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.105 Rms

class RmsCls

Rms commands group definition. 14 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.106 High

class HighCls

High commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.perror.rms.high.clone()
```

Subgroups

6.2.1.9.4.107 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪rms.high.average.calculate()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.high.
↪average.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.108 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪rms.high.current.calculate()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.high.
↪current.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.109 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪rms.high.extreme.calculate()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.high.
↪extreme.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.110 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:HIGH:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.high.
↪standardDev.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.9.4.111 Low

class LowCls

Low commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.modulation.perror.rms.low.clone()
```

Subgroups

6.2.1.9.4.112 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪rms.low.average.calculate()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.low.
↪average.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.113 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪rms.low.current.calculate()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.low.
↪current.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.114 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.perror.
↪rms.low.extreme.calculate()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.low.
↪extreme.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.115 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PERRor:RMS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PERRor:RMS:LOW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.perror.rms.low.
↪standardDev.fetch()
```

Return phase error RMS values for low and high EVM window position, 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.

Suppressed linked return values: reliability

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.9.4.116 Ppower

class PpowerCls

Ppower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.117 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.ppower.
↪average.calculate()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.ppower.average.
↪fetch()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.118 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.ppower.
↪current.calculate()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.ppower.current.
↪fetch()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.119 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MAXimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.ppower.
↪maximum.calculate()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MAXimum
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.ppower.maximum.
↪fetch()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.120 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MINimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.ppower.
↪minimum.calculate()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:MINimum
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.ppower.minimum.
↪fetch()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.121 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PPOWer:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.ppower.
↪standardDev.fetch()
```

Return user equipment peak 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.

Suppressed linked return values: reliability

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.122 Psd

class PsdCls

Psd commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.123 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.psd.
↪average.calculate()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.psd.average.
↪fetch()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: Comma-separated list of values, one per measured segment

6.2.1.9.4.124 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.psd.
↪current.calculate()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.psd.current.
↪fetch()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: Comma-separated list of values, one per measured segment

6.2.1.9.4.125 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:MAXimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.psd.
↪maximum.calculate()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MAXimum
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.psd.maximum.
↪fetch()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: Comma-separated list of values, one per measured segment

6.2.1.9.4.126 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:MINimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.psd.
↪minimum.calculate()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:MINimum
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.psd.minimum.
↪fetch()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: Comma-separated list of values, one per measured segment

6.2.1.9.4.127 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PSD:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:PSD:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.psd.standardDev.
↪fetch()
```

Return RB 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.

Suppressed linked return values: reliability

return
psd: Comma-separated list of values, one per measured segment

6.2.1.9.4.128 SchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SCHType
```

class SchTypeCls

SchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[SidelinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SCHType
value: List[enums.SidelinkChannelType] = driver.lteMeas.multiEval.listPy.
↪modulation.schType.fetch()
```

Returns the sidelink channel type evaluated for modulation results, for all measured list mode segments.

Suppressed linked return values: reliability

return
channel_type: Comma-separated list of values, one per measured segment

6.2.1.9.4.129 Terror

class TerrorCls

Terror commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.130 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:AVERage
value: List[float or bool] = driver.lteMeas.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.

Suppressed linked return values: reliability

return
timing_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:AVERage
value: List[float] = driver.lteMeas.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.

Suppressed linked return values: reliability

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.131 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:CURRent
value: List[float or bool] = driver.lteMeas.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.

Suppressed linked return values: reliability

return
timing_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:CURRent
value: List[float] = driver.lteMeas.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.

Suppressed linked return values: reliability

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.132 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.terror.
↪extreme.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.

Suppressed linked return values: reliability

return
timing_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.terror.extreme.
↪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.

Suppressed linked return values: reliability

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.133 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TERRor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TERRor:SDEViation
value: List[float] = driver.lteMeas.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.

Suppressed linked return values: reliability

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.9.4.134 Tpower

class TpowerCls

Tpower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.4.135 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.tpower.
↪average.calculate()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.tpower.average.
↪fetch()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.136 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.tpower.
↪current.calculate()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.tpower.current.
↪fetch()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.137 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MAXimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.tpower.
↪maximum.calculate()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MAXimum
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.tpower.maximum.
↪fetch()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.138 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MINimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.modulation.tpower.
↪minimum.calculate()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:MINimum
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.tpower.minimum.
↪fetch()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.4.139 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:TPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:MODulation:TPOWer:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.modulation.tpower.
↪standardDev.fetch()
```

Return user equipment 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.5 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.5.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:PEAK
value: List[float] = driver.lteMeas.multiEval.listPy.pmonitor.peak.fetch()
```

Return the power monitor results for all measured segments in list mode. The commands return one power result per subframe for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

INTRO_CMD_HELP: Commands for querying the result list structure:

- method RsCMPX_LteMeas.LteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Start.fetch
- method RsCMPX_LteMeas.LteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Length.fetch

Suppressed linked return values: reliability

return
step_peak_power: Comma-separated list of power values, one value per subframe, from first subframe of first measured segment to last subframe of last measured segment For an inactive segment only one INV is returned, independent of the number of configured subframes.

6.2.1.9.5.2 Rms

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:RMS
```

class RmsCls

Rms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:RMS
value: List[float] = driver.lteMeas.multiEval.listPy.pmonitor.rms.fetch()
```

Return the power monitor results for all measured segments in list mode. The commands return one power result per subframe for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

INTRO_CMD_HELP: Commands for querying the result list structure:

- method RsCMPX_LteMeas.LteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Start.fetch
- method RsCMPX_LteMeas.LteMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Length.fetch

Suppressed linked return values: reliability

return
step_rms_power: Comma-separated list of power values, one value per subframe, from first subframe of first measured segment to last subframe of last measured segment For an inactive segment only one INV is returned, independent of the number of configured subframes.

6.2.1.9.6 Power

class PowerCls

Power commands group definition. 9 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.6.1 TxPower

class TxPowerCls

TxPower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.6.2 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.power.txPower.
↪average.calculate()
```

No command help available

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.power.txPower.average.
↪fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.6.3 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.power.txPower.
↪current.calculate()
```

No command help available

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.power.txPower.current.
↪fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.6.4 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:MAXimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.power.txPower.
↪maximum.calculate()
```

No command help available

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
value: List[float] = driver.lteMeas.multiEval.listPy.power.txPower.maximum.
↪fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.6.5 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:MINimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.power.txPower.
↪minimum.calculate()
```

No command help available

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) No help available

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
value: List[float] = driver.lteMeas.multiEval.listPy.power.txPower.minimum.
↪fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.6.6 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.power.txPower.standardDev.
↪fetch()
```

Return the total TX power of all component carriers, for all measured list mode segments.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.7 Segment

RepCap Settings

```
# Range: Nr1 .. Nr2000
rc = driver.lteMeas.multiEval.listPy.segment.repcap_segment_get()
driver.lteMeas.multiEval.listPy.segment.repcap_segment_set(repcap.Segment.Nr1)
```

class SegmentCls

Segment commands group definition. 92 total commands, 7 Subgroups, 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.lteMeas.multiEval.listPy.segment.clone()
```

Subgroups

6.2.1.9.7.1 Aclr

class AclrCls

Aclr commands group definition. 7 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.2 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Utra_2_Neg: enums.ResultStatus2: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: enums.ResultStatus2: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: enums.ResultStatus2: ACLR for the first E-UTRA channel below the carrier frequency
- Eutra: enums.ResultStatus2: Power in the allocated E-UTRA channel
- Eutra_Positiv: enums.ResultStatus2: ACLR for the first E-UTRA channel above the carrier frequency
- Utra_1_Pos: enums.ResultStatus2: ACLR for the first UTRA channel with higher frequency
- Utra_2_Pos: enums.ResultStatus2: ACLR for the second UTRA channel with higher frequency

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Utra_2_Neg: float: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: float: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: float: ACLR for the first E-UTRA channel below the carrier frequency
- Eutra: float: Power in the allocated E-UTRA channel
- Eutra_Positiv: float: ACLR for the first E-UTRA channel above the carrier frequency
- Utra_1_Pos: float: ACLR for the first UTRA channel with higher frequency
- Utra_2_Pos: float: ACLR for the second UTRA channel with higher frequency

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.aclr.average.
↪calculate(segment = repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.aclr.average.
↪fetch(segment = repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.3 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Utra_2_Neg: enums.ResultStatus2: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: enums.ResultStatus2: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: enums.ResultStatus2: ACLR for the first E-UTRA channel below the carrier frequency
- Eutra: enums.ResultStatus2: Power in the allocated E-UTRA channel
- Eutra_Positiv: enums.ResultStatus2: ACLR for the first E-UTRA channel above the carrier frequency
- Utra_1_Pos: enums.ResultStatus2: ACLR for the first UTRA channel with higher frequency
- Utra_2_Pos: enums.ResultStatus2: ACLR for the second UTRA channel with higher frequency

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Utra_2_Neg: float: ACLR for the second UTRA channel with lower frequency
- Utra_1_Neg: float: ACLR for the first UTRA channel with lower frequency
- Eutra_Negativ: float: ACLR for the first E-UTRA channel below the carrier frequency
- Eutra: float: Power in the allocated E-UTRA channel
- Eutra_Positiv: float: ACLR for the first E-UTRA channel above the carrier frequency
- Utra_1_Pos: float: ACLR for the first UTRA channel with higher frequency
- Utra_2_Pos: float: ACLR for the second UTRA channel with higher frequency

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.aclr.current.
↪calculate(segment = repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.aclr.current.
↪fetch(segment = repcap.Segment.Default)
```

Return ACLR single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.4 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Nr_Res_Blocks: int: Number of allocated resource blocks
- Offset_Res_Blocks: int: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.aclr.dallocation.
↪fetch(segment = repcap.Segment.Default)
```

Return the detected allocation for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

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.

6.2.1.9.7.5 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Channel_Type: enums.UplinkChannelType: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:DCHType
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.aclr.dchType.
↪fetch(segment = repcap.Segment.Default)
```

Return the uplink channel type for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

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.

6.2.1.9.7.6 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:DMODulation
```

class DmodulationCls

Dmodulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Modulation: enums.Modulation: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:DMODulation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.aclr.dmodulation.
↪fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.7 EsFlatness

class EsFlatnessCls

EsFlatness commands group definition. 9 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.8 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.
↪average.calculate(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value 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 CalculateStruct structure arguments.

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.average.
↪fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Suppressed linked return values: reliability

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.

6.2.1.9.7.9 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 1 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: Reliability indicator for the segment

- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.
↪current.calculate(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value 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 CalculateStruct structure arguments.

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.current.
↪fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Suppressed linked return values: reliability

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.lteMeas.multiEval.listPy.segment.esFlatness.current.clone()
```

Subgroups

6.2.1.9.7.10 ScIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:CURRent:SCINdex
```

class ScIndexCls

ScIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Maximum_1: int: SC index of max (range 1)
- Minimum_1: int: SC index of min (range 1)
- Maximum_2: int: SC index of max (range 2)
- Minimum_2: int: SC index of min (range 2)

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:CURRent:SCINdex
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.current.
↪scIndex.fetch(segment = repcap.Segment.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for segment <no> in list mode. At these SC indices, the current minimum and maximum power of the equalizer coefficients have been detected within range 1 and range 2.

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.

6.2.1.9.7.11 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Ripple_1: float or bool: Limit check result for max (range 1) - min (range 1) .
- Ripple_2: float or bool: Limit check result for max (range 2) - min (range 2) .
- Max_R_1_Min_R_2: float or bool: Limit check result for max (range 1) - min (range 2) .
- Max_R_2_Min_R_1: float or bool: Limit check result for max (range 2) - min (range 1) .

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:EXTReme
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.
↪extreme.calculate(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value 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 CalculateStruct structure arguments.

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.extreme.
↪fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Suppressed linked return values: reliability

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.

6.2.1.9.7.12 StandardDev

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ESFLatness:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Ripple_1: float or bool: No parameter help available
- Ripple_2: float or bool: No parameter help available
- Max_R_1_Min_R_2: float or bool: No parameter help available
- Max_R_2_Min_R_1: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots

- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Ripple_1: float: Max (range 1) - min (range 1)
- Ripple_2: float: Max (range 2) - min (range 2)
- Max_R_1_Min_R_2: float: Max (range 1) - min (range 2)
- Max_R_2_Min_R_1: float: Max (range 2) - min (range 1)
- Min_R_1: float: Min (range 1)
- Max_R_1: float: Max (range 1)
- Min_R_2: float: Min (range 2)
- Max_R_2: float: Max (range 2)

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.
↪standardDev.calculate(segment = repcap.Segment.Default)
```

No command help available

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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ESFLatness:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.esFlatness.
↪standardDev.fetch(segment = repcap.Segment.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode.

Suppressed linked return values: reliability

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.

6.2.1.9.7.13 InbandEmission

class InbandEmissionCls

InbandEmission commands group definition. 18 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.14 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.repcap_carrierComponent_
↪get()
driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.repcap_carrierComponent_
↪set(repcap.CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.7.15 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.16 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.
↪margin.average.fetch(segment = repcap.Segment.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

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

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

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

6.2.1.9.7.17 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.
↪margin.current.fetch(segment = repcap.Segment.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

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

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

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.lteMeas.multiEval.listPy.segment.inbandEmission.cc.margin.current.clone()
```

Subgroups

6.2.1.9.7.18 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Rb_Index: int: Resource block index of margin

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.
↪margin.current.rbIndex.fetch(segment = repcap.Segment.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the component carrier CC<c> in-band emission measurement for segment <no> in list mode. At these RB indices, the CURRent and EXTReme margins have been detected.

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

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

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

6.2.1.9.7.19 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.
↪margin.extreme.fetch(segment = repcap.Segment.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

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

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

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.lteMeas.multiEval.listPy.segment.inbandEmission.cc.margin.extreme.clone()
```

Subgroups

6.2.1.9.7.20 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Rb_Index: int: Resource block index of margin

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.
↪margin.extreme.rbIndex.fetch(segment = repcap.Segment.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the component carrier CC<c> in-band emission measurement for segment <no> in list mode. At these RB indices, the CURRent and EXTReme margins have been detected.

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

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

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

6.2.1.9.7.21 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:CC<c>
↪:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:CC<c>:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.cc.
↪margin.standardDev.fetch(segment = repcap.Segment.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for component carrier CC<c>, segment <no> in list mode. The CURRent margins indicate the minimum (vertical) distance between the limit line and the current trace. A negative result indicates that the limit is exceeded. The AVERage, EXTReme and SDEViation values are calculated from the current margins.

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

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

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

6.2.1.9.7.22 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.23 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available

- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.
↪margin.average.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.24 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.
↪margin.current.fetch(segment = repcap.Segment.Default)
```

No command help available

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.lteMeas.multiEval.listPy.segment.inbandEmission.margin.current.clone()
```

Subgroups

6.2.1.9.7.25 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.
↪margin.current.rbIndex.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.26 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.
↪margin.extreme.fetch(segment = repcap.Segment.Default)
```

No command help available

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.lteMeas.multiEval.listPy.segment.inbandEmission.margin.extreme.clone()
```

Subgroups

6.2.1.9.7.27 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.
↪margin.extreme.rbIndex.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.28 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.
↪margin.standardDev.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.29 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.repcap_secondaryCC_get()
driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.2.1.9.7.30 Margin

class MarginCls

Margin commands group definition. 6 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.31 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.
↪margin.average.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.9.7.32 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.
↪margin.current.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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.lteMeas.multiEval.listPy.segment.inbandEmission.scc.margin.current.
↪clone()
```

Subgroups

6.2.1.9.7.33 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:CURRent:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:CURRent:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.
↪margin.current.rbIndex.fetch(segment = repcap.Segment.Default, secondaryCC =␣
↪repcap.SecondaryCC.Default)
```

No command help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.9.7.34 Extreme

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 1 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.
↪margin.extreme.fetch(segment = repcap.Segment.Default, secondaryCC = repcap.
↪SecondaryCC.Default)
```

No command help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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.lteMeas.multiEval.listPy.segment.inbandEmission.scc.margin.extreme.
↪clone()
```

Subgroups

6.2.1.9.7.35 RbIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:EXTReme:RBINdex
```

class RbIndexCls

RbIndex commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Rb_Index: int: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.
↪margin.extreme.rbIndex.fetch(segment = repcap.Segment.Default, secondaryCC =␣
↪repcap.SecondaryCC.Default)
```

No command help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.9.7.36 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:IEMission:SCC<c>
↪:MARGin:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin: float: No parameter help available

fetch(*segment=Segment.Default*, *secondaryCC=SecondaryCC.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:IEMission:SCC<c>:MARGin:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.inbandEmission.scc.
↪margin.standardDev.fetch(segment = repcap.Segment.Default, secondaryCC =␣
↪repcap.SecondaryCC.Default)
```

No command help available

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

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.9.7.37 Modulation

class ModulationCls

Modulation commands group definition. 17 total commands, 8 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.38 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage
```

class AverageCls

Average commands group definition. 7 total commands, 5 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position
- Iq_Offset: float or bool: I/Q origin offset
- Frequency_Error: float or bool: Carrier frequency error

- Timing_Error: float or bool: Time error
- Tx_Power: float or bool: User equipment power
- Peak_Power: float or bool: User equipment peak power
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float or bool: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float or bool: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float or bool: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float or bool: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float or bool: Phase error DMRS value, high EVM window position

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset
- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error
- Tx_Power: float: User equipment power
- Peak_Power: float: User equipment peak power
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position

- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.modulation.
↪average.calculate(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.average.
↪fetch(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.lteMeas.multiEval.listPy.segment.modulation.average.clone()
```

Subgroups

6.2.1.9.7.39 Dmrs

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage:DMRS
```

class DmrsCls

Dmrs commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Evm_Dmrs_Low: float: No parameter help available
- Evm_Dmrs_High: float: No parameter help available
- Mag_Err_Dmrs_Low: float: No parameter help available
- Mag_Err_Dmrs_High: float: No parameter help available
- Ph_Error_Dmrs_Low: float: No parameter help available
- Ph_Error_Dmrs_High: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage:DMRS
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.average.
↪dmrs.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.40 Emph

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage:EMPH
```

class EmphCls

Emph commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available

- Evm_Rms_Low: float: No parameter help available
- Evm_Rms_High: float: No parameter help available
- Evm_Peak_Low: float: No parameter help available
- Evm_Peak_High: float: No parameter help available
- Mag_Error_Rms_Low: float: No parameter help available
- Mag_Error_Rms_High: float: No parameter help available
- Mag_Error_Peak_Low: float: No parameter help available
- Mag_Err_Peak_High: float: No parameter help available
- Ph_Error_Rms_Low: float: No parameter help available
- Ph_Error_Rms_High: float: No parameter help available
- Ph_Error_Peak_Low: float: No parameter help available
- Ph_Error_Peak_High: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage:EMPH
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.average.
↪emph.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.41 Globale

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage:GLOBal
```

class GlobaleCls

Globale commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage:GLOBal
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.average.
↪globale.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.42 Mod

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage:MOD
```

class ModCls

Mod commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Iq_Offset: float: No parameter help available
- Frequency_Error: float: No parameter help available
- Timing_Error: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage:MOD
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.average.
↪mod.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.43 Pow

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:AVERage:POW
```

class PowCls

Pow commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Tx_Power: float: No parameter help available
- Peak_Power: float: No parameter help available
- Psd: float: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:AVERage:POW
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.average.
↪pow.fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.44 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position

- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position
- Iq_Offset: float or bool: I/Q origin offset
- Frequency_Error: float or bool: Carrier frequency error
- Timing_Error: float or bool: Time error
- Tx_Power: float or bool: User equipment power
- Peak_Power: float or bool: User equipment peak power
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float or bool: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float or bool: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float or bool: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float or bool: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float or bool: Phase error DMRS value, high EVM window position

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position

- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset
- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error
- Tx_Power: float: User equipment power
- Peak_Power: float: User equipment peak power
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.modulation.
↪current.calculate(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.current.
↪fetch(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.45 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Nr_Res_Blocks: int: Number of allocated resource blocks
- Offset_Res_Blocks: int: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.
↪dallocation.fetch(segment = repcap.Segment.Default)
```

Return the detected allocation for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

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.

6.2.1.9.7.46 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Channel_Type: enums.UplinkChannelType: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:DCHType
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.dchType.
↪fetch(segment = repcap.Segment.Default)
```

Return the uplink channel type for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

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.

6.2.1.9.7.47 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:DMODulation
```

class DmodulationCls

Dmodulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Modulation: enums.Modulation: QPSK, 16QAM, 64QAM, 256QAM

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:DMODulation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.
↪dmodulation.fetch(segment = repcap.Segment.Default)
```

Return the detected modulation scheme for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. If channel type PUCCH is detected, QPSK is returned for the modulation scheme because the QPSK limits are applied in that case.

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.

6.2.1.9.7.48 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position

- Iq_Offset: float or bool: I/Q origin offset
- Frequency_Error: float or bool: Carrier frequency error
- Timing_Error: float or bool: Time error
- Tx_Power_Minimum: float or bool: Minimum user equipment power
- Tx_Power_Maximum: float or bool: Maximum user equipment power
- Peak_Power_Min: float or bool: Minimum user equipment peak power
- Peak_Power_Max: float or bool: Maximum user equipment peak power
- Psd_Minimum: float or bool: No parameter help available
- Psd_Maximum: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float or bool: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float or bool: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float or bool: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float or bool: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float or bool: Phase error DMRS value, high EVM window position

`class FetchStruct`

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset
- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error

- Tx_Power_Minimum: float: Minimum user equipment power
- Tx_Power_Maximum: float: Maximum user equipment power
- Peak_Power_Min: float: Minimum user equipment peak power
- Peak_Power_Max: float: Maximum user equipment peak power
- Psd_Minimum: float: No parameter help available
- Psd_Maximum: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:EXTReme
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.modulation.
↪extreme.calculate(segment = repcap.Segment.Default)
```

Return modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.extreme.
↪fetch(segment = repcap.Segment.Default)
```

Return modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.49 SchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:SCHType
```

class SchTypeCls

SchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Channel_Type: enums.SidelinkChannelType: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:SCHType
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.schType.
↪fetch(segment = repcap.Segment.Default)
```

Returns the sidelink channel type evaluated for modulation 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.

6.2.1.9.7.50 StandardDev

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:MODulation:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Evm_Rms_Low: float or bool: No parameter help available
- Evm_Rms_High: float or bool: No parameter help available

- Evm_Peak_Low: float or bool: No parameter help available
- Evm_Peak_High: float or bool: No parameter help available
- Mag_Error_Rms_Low: float or bool: No parameter help available
- Mag_Error_Rms_High: float or bool: No parameter help available
- Mag_Error_Peak_Low: float or bool: No parameter help available
- Mag_Err_Peak_High: float or bool: No parameter help available
- Ph_Error_Rms_Low: float or bool: No parameter help available
- Ph_Error_Rms_High: float or bool: No parameter help available
- Ph_Error_Peak_Low: float or bool: No parameter help available
- Ph_Error_Peak_High: float or bool: No parameter help available
- Iq_Offset: float or bool: No parameter help available
- Frequency_Error: float or bool: No parameter help available
- Timing_Error: float or bool: No parameter help available
- Tx_Power: float or bool: No parameter help available
- Peak_Power: float or bool: No parameter help available
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: No parameter help available
- Evm_Dmrs_High: float or bool: No parameter help available
- Mag_Err_Dmrs_Low: float or bool: No parameter help available
- Mag_Err_Dmrs_High: float or bool: No parameter help available
- Ph_Error_Dmrs_Low: float or bool: No parameter help available
- Ph_Error_Dmrs_High: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position

- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset
- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error
- Tx_Power: float: User equipment power
- Peak_Power: float: User equipment peak power
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.modulation.
↪standardDev.calculate(segment = repcap.Segment.Default)
```

No command help available

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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:MODulation:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.modulation.
↪standardDev.fetch(segment = repcap.Segment.Default)
```

Returns modulation single-value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.51 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 4 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.52 Array

class ArrayCls

Array commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.53 Length

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:ARRay:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*segment=Segment.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:ARRay:LENGth
value: int = driver.lteMeas.multiEval.listPy.segment.pmonitor.array.length.
↪fetch(segment = repcap.Segment.Default)
```

Returns the number of power monitor results for segment <no> contained in a result list for all measured segments. Such a result list is, for example, returned by the command method RsCMPX_LteMeas.LteMeas.MultiEval.ListPy.Pmonitor.Rms.fetch.

Suppressed linked return values: reliability

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

return
length: Number of power monitor results

6.2.1.9.7.54 Start

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:ARRay:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*segment=Segment.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:ARRay:STARt
value: int = driver.lteMeas.multiEval.listPy.segment.pmonitor.array.start.
↪fetch(segment = repcap.Segment.Default)
```

Returns the offset of the first power monitor result for segment <no> within a result list for all measured segments. Such a result list is, for example, returned by the command method RsCMPX_LteMeas.LteMeas.MultiEval.ListPy.Pmonitor.Rms. fetch. A returned <Start> value n indicates that the result for the first subframe of the segment is the (n+1) th result in the power result list over all segments.

Suppressed linked return values: reliability

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

return
start: Offset of the first power monitor result

6.2.1.9.7.55 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment

- Step_Peak_Power: List[float]: Comma-separated list of power values, one value per subframe, from first to last subframe of the segment. For an inactive segment, only one INV is returned, independent of the number of configured subframes.

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:PEAK
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.pmonitor.peak.
↪fetch(segment = repcap.Segment.Default)
```

Return the power monitor results for segment <no> in list mode. The commands return one power result for each subframe of the segment for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

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.

6.2.1.9.7.56 Rms

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:RMS
```

class RmsCls

Rms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Step_Rms_Power: List[float]: Comma-separated list of power values, one value per subframe, from first to last subframe of the segment. For an inactive segment, only one INV is returned, independent of the number of configured subframes.

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:RMS
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.pmonitor.rms.
↪fetch(segment = repcap.Segment.Default)
```

Return the power monitor results for segment <no> in list mode. The commands return one power result for each subframe of the segment for the measured carrier. The power values are RMS averaged over the subframe or represent the peak value within the subframe.

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.

6.2.1.9.7.57 Power

class PowerCls

Power commands group definition. 18 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.58 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power: float: Total TX power of all component carriers

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.average.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.average.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX 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.

6.2.1.9.7.59 Cc<CarrierComponentB>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.listPy.segment.power.cc.repcap_carrierComponentB_get()
driver.lteMeas.multiEval.listPy.segment.power.cc.repcap_carrierComponentB_set(repcap.
↪CarrierComponentB.Nr1)
```

class CcCls

Cc commands group definition. 9 total commands, 5 Subgroups, 0 group commands Repeated Capability: CarrierComponentB, default value after init: CarrierComponentB.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.7.60 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power: float: TX power of the component carrier

calculate(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.
↪average.calculate(segment = repcap.Segment.Default, carrierComponentB =␣
↪repcap.CarrierComponentB.Default)
```

No command help available

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

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

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

fetch(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.average.
↪fetch(segment = repcap.Segment.Default, carrierComponentB = repcap.
↪CarrierComponentB.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

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

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

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

6.2.1.9.7.61 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power: float: TX power of the component carrier

calculate(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.
↪current.calculate(segment = repcap.Segment.Default, carrierComponentB =␣
↪repcap.CarrierComponentB.Default)
```

No command help available

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

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

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

fetch(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.current.
↪fetch(segment = repcap.Segment.Default, carrierComponentB = repcap.
↪CarrierComponentB.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

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

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

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

6.2.1.9.7.62 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available

- Tx_Power_Max: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power_Max: float: TX power of the component carrier

calculate(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.
↪maximum.calculate(segment = repcap.Segment.Default, carrierComponentB =␣
↪repcap.CarrierComponentB.Default)
```

No command help available

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

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

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

fetch(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:MAXimum
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.maximum.
↪fetch(segment = repcap.Segment.Default, carrierComponentB = repcap.
↪CarrierComponentB.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

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

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

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

6.2.1.9.7.63 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power_Min: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power_Min: float: TX power of the component carrier

calculate(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CC<no>:MINimum
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.
↪minimum.calculate(segment = repcap.Segment.Default, carrierComponentB =␣
↪repcap.CarrierComponentB.Default)
```

No command help available

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

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

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

fetch(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:MINimum
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.minimum.
↪fetch(segment = repcap.Segment.Default, carrierComponentB = repcap.
↪CarrierComponentB.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

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

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

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

6.2.1.9.7.64 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC<no>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power: float: TX power of the component carrier

fetch(*segment=Segment.Default*, *carrierComponentB=CarrierComponentB.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CC
↪<no>:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.cc.
↪standardDev.fetch(segment = repcap.Segment.Default, carrierComponentB =␣
↪repcap.CarrierComponentB.Default)
```

Return TX power results for component carrier CC<no> and a single segment in list mode.

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

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

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

6.2.1.9.7.65 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power: float: Total TX power of all component carriers

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.current.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.current.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX 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.

6.2.1.9.7.66 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power_Max: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power_Max: float: Total TX power of all component carriers

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.maximum.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MAXimum
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.maximum.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX 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.

6.2.1.9.7.67 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power_Min: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power_Min: float: Total TX power of all component carriers

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MINimum
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.power.minimum.
↪calculate(segment = repcap.Segment.Default)
```

No command help available

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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:MINimum
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.minimum.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX 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.

6.2.1.9.7.68 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:POWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in subframes
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Tx_Power: float: Total TX power of all component carriers

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:POWer:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.power.standardDev.
↪fetch(segment = repcap.Segment.Default)
```

Return total TX 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.

6.2.1.9.7.69 SeMask

class SeMaskCls

SeMask commands group definition. 19 total commands, 8 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.70 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Obw: float or bool: Occupied bandwidth
- Tx_Power: float or bool: Total TX power in the slot

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.seMask.average.
↪calculate(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.average.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.71 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Obw: float or bool: Occupied bandwidth
- Tx_Power: float or bool: Total TX power in the slot

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.seMask.current.
↪calculate(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.current.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.72 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Nr_Res_Blocks: int: Number of allocated resource blocks
- Offset_Res_Blocks: int: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.dallocation.
↪fetch(segment = repcap.Segment.Default)
```

Return the detected allocation for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

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.

6.2.1.9.7.73 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Channel_Type: enums.UplinkChannelType: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:DCHType
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.dchType.
↪fetch(segment = repcap.Segment.Default)
```

Return the uplink channel type for segment <no> in list mode. The result is determined from the last measured slot of the statistical length. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

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.

6.2.1.9.7.74 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:DMODulation
```

class DmodulationCls

Dmodulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available

- Modulation: enums.Modulation: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:DMODulation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.dmodulation.
↪fetch(segment = repcap.Segment.Default)
```

No command help available

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.

6.2.1.9.7.75 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Obw: float or bool: Occupied bandwidth
- Tx_Power_Min: float or bool: Minimum total TX power in the slot
- Tx_Power_Max: float or bool: Maximum total TX power in the slot

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Obw: float: Occupied bandwidth
- Tx_Power_Min: float: Minimum total TX power in the slot
- Tx_Power_Max: float: Maximum total TX power in the slot

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:EXTReme
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.seMask.extreme.
↪calculate(segment = repcap.Segment.Default)
```

Return spectrum emission extreme results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:EXTReme
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.extreme.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission extreme results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.7.76 Margin

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:MARGin
```

class MarginCls

Margin commands group definition. 8 total commands, 4 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available

- Margin_Curr_Neg: List[float]: No parameter help available
- Margin_Curr_Pos: List[float]: No parameter help available
- Margin_Avg_Neg: List[float]: No parameter help available
- Margin_Avg_Pos: List[float]: No parameter help available
- Margin_Min_Neg: List[float]: No parameter help available
- Margin_Min_Pos: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.
↪fetch(segment = repcap.Segment.Default)
```

No command help available

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.lteMeas.multiEval.listPy.segment.seMask.margin.clone()
```

Subgroups

6.2.1.9.7.77 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:MARGin:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin_Curr_Neg: List[float]: No parameter help available
- Margin_Curr_Pos: List[float]: No parameter help available

- Margin_Avg_Neg: List[float]: No parameter help available
- Margin_Avg_Pos: List[float]: No parameter help available
- Margin_Min_Neg: List[float]: No parameter help available
- Margin_Min_Pos: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:ALL
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.all.
↪fetch(segment = repcap.Segment.Default)
```

Return limit line margin values, i.e. vertical distances between the spectrum emission mask and a trace, 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.

6.2.1.9.7.78 Average

class AverageCls

Average commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.79 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:NEGativ
```

class NegativCls

Negativ commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment

- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin_Avg_Neg_X: List[float]: No parameter help available
- Margin_Avg_Neg_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:NEGativ
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.
↪average.negativ.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {...}area12

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.

6.2.1.9.7.80 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:POSitiv
```

class PositivCls

Positiv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin_Avg_Pos_X: List[float]: No parameter help available
- Margin_Avg_Pos_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:AVERage:POSitiv
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.
↪average.positiv.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <Statist-Expired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {... }area12

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.

6.2.1.9.7.81 Current

class CurrentCls

Current commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.82 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:NEGativ
```

class NegativCls

Negativ commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin_Curr_Neg_X: List[float]: No parameter help available
- Margin_Curr_Neg_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:NEGativ
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.
↪current.negativ.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <Statist-Expired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {...}area12

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.

6.2.1.9.7.83 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:POSitiv
```

class PositivCls

Positiv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin_Curr_Pos_X: List[float]: No parameter help available
- Margin_Curr_Pos_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:CURRent:POSitiv
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.
↪current.positiv.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <Statist-Expired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {...}area12

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.

6.2.1.9.7.84 Minimum

class MinimumCls

Minimum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.7.85 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:NEGativ
```

class NegativCls

Negativ commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin_Min_Neg_X: List[float]: No parameter help available
- Margin_Min_Neg_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:NEGativ
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.
↪minimum.negativ.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {... }area12

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.

6.2.1.9.7.86 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:POSitiv
```

class PositivCls

Positiv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Margin_Min_Pos_X: List[float]: No parameter help available
- Margin_Min_Pos_Y: List[float]: No parameter help available

fetch(*segment=Segment.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:MARGin:MINimum:POSitiv
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.margin.
↪minimum.positiv.fetch(segment = repcap.Segment.Default)
```

Return spectrum emission mask margin results for segment <no> in list mode. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. Returned sequence: <Reliability>, <SegReliability>, <StatistExpired>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {... }area2, ..., {... }area12

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.

6.2.1.9.7.87 StandardDev

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:SEMask:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Seg_Reliability: int: No parameter help available
- Statist_Expired: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Obw: float or bool: No parameter help available
- Tx_Power: float or bool: No parameter help available

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Seg_Reliability: int: Reliability indicator for the segment
- Statist_Expired: int: Reached statistical length in slots
- Out_Of_Tolerance: int: Percentage of measured subframes with failed limit check
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot

calculate(*segment=Segment.Default*) → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.listPy.segment.seMask.
↪standardDev.calculate(segment = repcap.Segment.Default)
```

No command help available

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*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:SEMask:SDEViation
value: FetchStruct = driver.lteMeas.multiEval.listPy.segment.seMask.standardDev.
↪fetch(segment = repcap.Segment.Default)
```

Return spectrum emission single value results for segment <no> in list mode. The values described below are returned by FETCh commands. The first four 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.

6.2.1.9.8 SeMask

class SeMaskCls

SeMask commands group definition. 24 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.8.1 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Nr_Res_Blocks: List[int]: Number of allocated resource blocks
- Offset_Res_Blocks: List[int]: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.listPy.seMask.dallocation.fetch()
```

Return the detected allocation for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ. The results are returned as pairs per segment: <Reliability>, {<NrResBlocks>, <OffsetResBlocks>}Seg 1, {<NrResBlocks>, <OffsetResBlocks>}Seg 2, ...

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

6.2.1.9.8.2 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[UplinkChannelType]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:DCHType
value: List[enums.UplinkChannelType] = driver.lteMeas.multiEval.listPy.seMask.
↪dchType.fetch()
```

Return the uplink channel type for all measured list mode segments. The result is determined from the last measured slot of the statistical length of a segment. The individual measurements provide the same result when measuring the same slot. However different statistical lengths can be defined for the measurements so that the measured slots and returned results can differ.

Suppressed linked return values: reliability

return
channel_type: Comma-separated list of values, one per measured segment

6.2.1.9.8.3 Margin

class MarginCls

Margin commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.8.4 Area<Area>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.lteMeas.multiEval.listPy.seMask.margin.area.repcap_area_get()
driver.lteMeas.multiEval.listPy.seMask.margin.area.repcap_area_set(repcap.Area.Nr1)
```

class AreaCls

Area commands group definition. 6 total commands, 2 Subgroups, 0 group commands Repeated Capability: Area, default value after init: Area.Nr1

Cloning the Group

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

Subgroups

6.2.1.9.8.5 Negativ

class NegativCls

Negativ commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.seMask.margin.area.negativ.clone()
```

Subgroups

6.2.1.9.8.6 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:NEGativ:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Margin_Avg_Neg_X: List[float]: No parameter help available

- Margin_Avg_Neg_Y: List[float]: No parameter help available

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:NEGativ:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.seMask.margin.area.negativ.
↪average.fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

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

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

6.2.1.9.8.7 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:NEGativ:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Margin_Curr_Neg_X: List[float]: No parameter help available
- Margin_Curr_Neg_Y: List[float]: No parameter help available

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:NEGativ:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.seMask.margin.area.negativ.
↪current.fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

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

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

6.2.1.9.8.8 Minimum

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:NEGativ:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Margin_Min_Neg_X: List[float]: No parameter help available
- Margin_Min_Neg_Y: List[float]: No parameter help available

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:NEGativ:MINimum
value: FetchStruct = driver.lteMeas.multiEval.listPy.seMask.margin.area.negativ.
↪minimum.fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

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

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

6.2.1.9.8.9 Positiv

class PositivCls

Positiv commands group definition. 3 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.listPy.seMask.margin.area.positiv.clone()
```

Subgroups

6.2.1.9.8.10 Average

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:POSitiv:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Margin_Avg_Pos_X: List[float]: X-position of margin for selected area
- Margin_Avg_Pos_Y: List[float]: Y-value of margin for selected area

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:POSitiv:AVERage
value: FetchStruct = driver.lteMeas.multiEval.listPy.seMask.margin.area.positiv.
↪average.fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

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

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

6.2.1.9.8.11 Current

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:POSitiv:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Margin_Curr_Pos_X: List[float]: X-position of margin for selected area
- Margin_Curr_Pos_Y: List[float]: Y-value of margin for selected area

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:POSitiv:CURRent
value: FetchStruct = driver.lteMeas.multiEval.listPy.seMask.margin.area.positiv.
↪current.fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

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

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

6.2.1.9.8.12 Minimum

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>:POSitiv:MINimum
```

class MinimumCls

Minimum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Margin_Min_Pos_X: List[float]: X-position of margin for selected area
- Margin_Min_Pos_Y: List[float]: Y-value of margin for selected area

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr>
↪:POSitiv:MINimum
value: FetchStruct = driver.lteMeas.multiEval.listPy.seMask.margin.area.positiv.
↪minimum.fetch(area = repcap.Area.Default)
```

Return spectrum emission mask margin positions for all measured list mode segments. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) for NEGative and POSitive offset frequencies. The results are returned as pairs per segment: <Reliability>, {<MarginPosX>, <MarginPosY>}Seg 1, {<MarginPosX>, <MarginPosY>}Seg 2, ...

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

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

6.2.1.9.8.13 Obw

class ObwCls

Obw commands group definition. 7 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.8.14 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.seMask.obw.average.
↪calculate()
```

Return the occupied bandwidth 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.

Suppressed linked return values: reliability

return
obw: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.obw.average.fetch()
```

Return the occupied bandwidth 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.

Suppressed linked return values: reliability

return
obw: Comma-separated list of values, one per measured segment

6.2.1.9.8.15 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.seMask.obw.current.
↪calculate()
```

Return the occupied bandwidth 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.

Suppressed linked return values: reliability

return
obw: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.obw.current.fetch()
```

Return the occupied bandwidth 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.

Suppressed linked return values: reliability

return
obw: Comma-separated list of values, one per measured segment

6.2.1.9.8.16 Extreme

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
value: List[float or bool] = driver.lteMeas.multiEval.listPy.seMask.obw.extreme.
↪calculate()
```

Return the occupied bandwidth 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.

Suppressed linked return values: reliability

return
obw: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.obw.extreme.fetch()
```

Return the occupied bandwidth 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.

Suppressed linked return values: reliability

return
obw: Comma-separated list of values, one per measured segment

6.2.1.9.8.17 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.obw.standardDev.
↪fetch()
```

Return the occupied bandwidth 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.

Suppressed linked return values: reliability

return
obw: Comma-separated list of values, one per measured segment

6.2.1.9.8.18 TxPower

class TxPowerCls

TxPower commands group definition. 9 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.9.8.19 Average

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:AVERage
value: List[float or bool] = driver.lteMeas.multiEval.listPy.seMask.txPower.
↪average.calculate()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:AVERage
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.txPower.average.
↪fetch()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.8.20 Current

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:CURRent
value: List[float or bool] = driver.lteMeas.multiEval.listPy.seMask.txPower.
↪current.calculate()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:CURRent
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.txPower.current.
↪fetch()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.8.21 Maximum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MAXimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.seMask.txPower.
↪maximum.calculate()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MAXimum
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.txPower.maximum.
↪fetch()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.8.22 Minimum

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate() → List[float]

```
# SCPI: CALCulate:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MINimum
value: List[float or bool] = driver.lteMeas.multiEval.listPy.seMask.txPower.
↪minimum.calculate()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MINimum
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.txPower.minimum.
↪fetch()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.8.23 StandardDev

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:SDEViation
value: List[float] = driver.lteMeas.multiEval.listPy.seMask.txPower.standardDev.
↪fetch()
```

Return the total TX power in the slot 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.

Suppressed linked return values: reliability

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.9.9 Sreliability

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SRELiability
```

class SreliabilityCls

Sreliability commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SRELiability
value: List[int] = driver.lteMeas.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.

Suppressed linked return values: reliability

return
seg_reliability: 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.

6.2.1.10 Merror

class MerrorCls

Merror commands group definition. 15 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.10.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
```

class AverageCls

Average commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: Magnitude error value for low EVM window position
- High: List[float]: Magnitude error value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.merror.average.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
value: ResultData = driver.lteMeas.multiEval.merror.average.fetch()
```

Returns the values of the magnitude error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'Square Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage
value: ResultData = driver.lteMeas.multiEval.merror.average.read()
```

Returns the values of the magnitude error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'Square Magnitude Error, Phase Error'.

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.lteMeas.multiEval.merror.average.clone()
```

Subgroups

6.2.1.10.1.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage:NREF
value: ResultData = driver.lteMeas.multiEval.merror.average.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage:NREF
value: ResultData = driver.lteMeas.multiEval.merror.average.nref.read()
```

No command help available

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

6.2.1.10.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
```

class CurrentCls

Current commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: Magnitude error value for low EVM window position
- High: List[float]: Magnitude error value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.merror.current.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
value: ResultData = driver.lteMeas.multiEval.merror.current.fetch()
```

Returns the values of the magnitude error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, … See also 'Square Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent
value: ResultData = driver.lteMeas.multiEval.merror.current.read()
```

Returns the values of the magnitude error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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.lteMeas.multiEval.merror.current.clone()
```

Subgroups

6.2.1.10.2.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent:NREF
value: ResultData = driver.lteMeas.multiEval.merror.current.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent:NREF
value: ResultData = driver.lteMeas.multiEval.merror.current.nref.read()
```

No command help available

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

6.2.1.10.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: Magnitude error value for low EVM window position
- High: List[float]: Magnitude error value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.merror.maximum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
value: ResultData = driver.lteMeas.multiEval.merror.maximum.fetch()
```

Returns the values of the magnitude error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum
value: ResultData = driver.lteMeas.multiEval.merror.maximum.read()
```

Returns the values of the magnitude error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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.lteMeas.multiEval.merror.maximum.clone()
```

Subgroups

6.2.1.10.3.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available

- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum:NREF
value: ResultData = driver.lteMeas.multiEval.merror.maximum.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum:NREF
value: ResultData = driver.lteMeas.multiEval.merror.maximum.nref.read()
```

No command help available

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

6.2.1.11 Modulation

class ModulationCls

Modulation commands group definition. 16 total commands, 8 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.11.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position
- Iq_Offset: float or bool: I/Q origin offset
- Frequency_Error: float or bool: Carrier frequency error
- Timing_Error: float or bool: Time error
- Tx_Power: float or bool: User equipment power
- Peak_Power: float or bool: User equipment peak power
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float or bool: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float or bool: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float or bool: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float or bool: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float or bool: Phase error DMRS value, high EVM window position
- Iq_Gain_Imbalance: float or bool: Gain imbalance
- Iq_Quadrature_Err: float or bool: Quadrature error
- Evm_Srs: float: Error vector magnitude for SRS signals

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.

- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset
- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error
- Tx_Power: float: User equipment power
- Peak_Power: float: User equipment peak power
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position
- Iq_Gain_Imbalance: float: Gain imbalance
- Iq_Quadrature_Err: float: Quadrature error
- Evm_Srs: float: Error vector magnitude for SRS signals

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.modulation.average.calculate()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.lteMeas.multiEval.modulation.average.fetch()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage
value: ResultData = driver.lteMeas.multiEval.modulation.average.read()
```

Return the current, average and standard deviation single-value results. 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.

6.2.1.11.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position

- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position
- Iq_Offset: float or bool: I/Q origin offset
- Frequency_Error: float or bool: Carrier frequency error
- Timing_Error: float or bool: Time error
- Tx_Power: float or bool: User equipment power
- Peak_Power: float or bool: User equipment peak power
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float or bool: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float or bool: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float or bool: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float or bool: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float or bool: Phase error DMRS value, high EVM window position
- Iq_Gain_Imbalance: float or bool: Gain imbalance
- Iq_Quadrature_Err: float or bool: Quadrature error
- Evm_Srs: float: Error vector magnitude for SRS signals

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset
- Frequency_Error: float: Carrier frequency error

- Timing_Error: float: Time error
- Tx_Power: float: User equipment power
- Peak_Power: float: User equipment peak power
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position
- Iq_Gain_Imbalance: float: Gain imbalance
- Iq_Quadrature_Err: float: Quadrature error
- Evm_Srs: float: Error vector magnitude for SRS signals

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.modulation.current.calculate()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.lteMeas.multiEval.modulation.current.fetch()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent
value: ResultData = driver.lteMeas.multiEval.modulation.current.read()
```

Return the current, average and standard deviation single-value results. 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.

6.2.1.11.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Nr_Res_Blocks: int: Number of allocated resource blocks
- Offset_Res_Blocks: int: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.modulation.dallocation.fetch()
```

Returns the detected allocation for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

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

6.2.1.11.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → UplinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DCHType
value: enums.UplinkChannelType = driver.lteMeas.multiEval.modulation.dchType.
↪fetch()
```

Returns the uplink channel type for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.11.5 Dmodulation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DMODulation
```

class DmodulationCls

Dmodulation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → Modulation

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:DMODulation
value: enums.Modulation = driver.lteMeas.multiEval.modulation.dmodulation.
↪fetch()
```

Returns the detected modulation scheme in the measured slot. If channel type PUCCH is detected, QPSK is returned for the modulation scheme because the QPSK limits are applied in that case.

Suppressed linked return values: reliability

return
modulation: QPSK, 16QAM, 64QAM, 256QAM

6.2.1.11.6 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
```

class ExtremeCls

Extreme commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position

- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position
- Iq_Offset: float or bool: I/Q origin offset
- Frequency_Error: float or bool: Carrier frequency error
- Timing_Error: float or bool: Time error
- Tx_Power_Minimum: float or bool: Minimum user equipment power
- Tx_Power_Maximum: float or bool: Maximum user equipment power
- Peak_Power_Min: float or bool: Minimum user equipment peak power
- Peak_Power_Max: float or bool: Maximum user equipment peak power
- Psd_Minimum: float or bool: No parameter help available
- Psd_Maximum: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float or bool: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float or bool: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float or bool: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float or bool: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float or bool: Phase error DMRS value, high EVM window position
- Iq_Gain_Imbalance: float or bool: Gain imbalance
- Iq_Quadrature_Err: float or bool: Quadrature error
- Evm_Srs: float: Error vector magnitude for SRS signals

class ResultData

Response structure. Fields:

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset

- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error
- Tx_Power_Minimum: float: Minimum user equipment power
- Tx_Power_Maximum: float: Maximum user equipment power
- Peak_Power_Min: float: Minimum user equipment peak power
- Peak_Power_Max: float: Maximum user equipment peak power
- Psd_Minimum: float: No parameter help available
- Psd_Maximum: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position
- Iq_Gain_Imbalance: float: Gain imbalance
- Iq_Quadrature_Err: float: Quadrature error
- Evm_Srs: float: Error vector magnitude for SRS signals

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
value: CalculateStruct = driver.lteMeas.multiEval.modulation.extreme.calculate()
```

Return the extreme single value results. 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.

Suppressed linked return values: reliability

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
value: ResultData = driver.lteMeas.multiEval.modulation.extreme.fetch()
```

Return the extreme single value results. 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.

Suppressed linked return values: reliability

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme
value: ResultData = driver.lteMeas.multiEval.modulation.extreme.read()
```

Return the extreme single value results. 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.

Suppressed linked return values: reliability

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

6.2.1.11.7 SchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SCHType
```

class SchTypeCls

SchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → SidelinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SCHType
value: enums.SidelinkChannelType = driver.lteMeas.multiEval.modulation.schType.
↪fetch()
```

Returns the sidelink channel type evaluated for modulation results.

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.11.8 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Evm_Rms_Low: float or bool: No parameter help available
- Evm_Rms_High: float or bool: No parameter help available
- Evm_Peak_Low: float or bool: No parameter help available
- Evm_Peak_High: float or bool: No parameter help available
- Mag_Error_Rms_Low: float or bool: No parameter help available

- Mag_Error_Rms_High: float or bool: No parameter help available
- Mag_Error_Peak_Low: float or bool: No parameter help available
- Mag_Err_Peak_High: float or bool: No parameter help available
- Ph_Error_Rms_Low: float or bool: No parameter help available
- Ph_Error_Rms_High: float or bool: No parameter help available
- Ph_Error_Peak_Low: float or bool: No parameter help available
- Ph_Error_Peak_High: float or bool: No parameter help available
- Iq_Offset: float or bool: No parameter help available
- Frequency_Error: float or bool: No parameter help available
- Timing_Error: float or bool: No parameter help available
- Tx_Power: float or bool: No parameter help available
- Peak_Power: float or bool: No parameter help available
- Psd: float or bool: No parameter help available
- Evm_Dmrs_Low: float or bool: No parameter help available
- Evm_Dmrs_High: float or bool: No parameter help available
- Mag_Err_Dmrs_Low: float or bool: No parameter help available
- Mag_Err_Dmrs_High: float or bool: No parameter help available
- Ph_Error_Dmrs_Low: float or bool: No parameter help available
- Ph_Error_Dmrs_High: float or bool: No parameter help available
- Iq_Gain_Imbalance: float or bool: No parameter help available
- Iq_Quadrature_Err: float or bool: No parameter help available
- Evm_Srs: float: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position

- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Iq_Offset: float: I/Q origin offset
- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error
- Tx_Power: float: User equipment power
- Peak_Power: float: User equipment peak power
- Psd: float: No parameter help available
- Evm_Dmrs_Low: float: EVM DMRS value, low EVM window position
- Evm_Dmrs_High: float: EVM DMRS value, high EVM window position
- Mag_Err_Dmrs_Low: float: Magnitude error DMRS value, low EVM window position
- Mag_Err_Dmrs_High: float: Magnitude error DMRS value, high EVM window position
- Ph_Error_Dmrs_Low: float: Phase error DMRS value, low EVM window position
- Ph_Error_Dmrs_High: float: Phase error DMRS value, high EVM window position
- Iq_Gain_Imbalance: float: Gain imbalance
- Iq_Quadrature_Err: float: Quadrature error
- Evm_Srs: float: Error vector magnitude for SRS signals

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.modulation.standardDev.
↪calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.lteMeas.multiEval.modulation.standardDev.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation
value: ResultData = driver.lteMeas.multiEval.modulation.standardDev.read()
```

Return the current, average and standard deviation single-value results. 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.

6.2.1.12 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 15 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.12.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available

- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.pdynamics.average.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
value: ResultData = driver.lteMeas.multiEval.pdynamics.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:AVERage
value: ResultData = driver.lteMeas.multiEval.pdynamics.average.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)

- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.1.12.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.pdynamics.current.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
value: ResultData = driver.lteMeas.multiEval.pdynamics.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:CURRent
value: ResultData = driver.lteMeas.multiEval.pdynamics.current.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.1.12.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.pdynamics.maximum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
value: ResultData = driver.lteMeas.multiEval.pdynamics.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MAXimum
value: ResultData = driver.lteMeas.multiEval.pdynamics.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.1.12.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
value: CalculateStruct = driver.lteMeas.multiEval.pdynamics.minimum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
value: ResultData = driver.lteMeas.multiEval.pdynamics.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:MINimum
value: ResultData = driver.lteMeas.multiEval.pdynamics.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.1.12.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Off_Power_Before: float: No parameter help available
- On_Power_Rms: float: No parameter help available
- On_Power_Peak: float: No parameter help available
- Off_Power_After: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.pdynamics.standardDev.
↪calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
value: ResultData = driver.lteMeas.multiEval.pdynamics.standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)
- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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:LTE:MEASurement<Instance>:MEValuation:PDYNamics:SDEViation
value: ResultData = driver.lteMeas.multiEval.pdynamics.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics measurement. A single result table row is returned, from left to right. The meaning of the values depends on the selected time mask, as follows:

Table Header: Time mask / <Power1> / <Power2> / <Power3> / <Power4>

- General on / off / OFF power (before) / ON power RMS / ON power peak / OFF power (after)

- PUCCH / PUSCH / SRS / SRS ON / ON power RMS / ON power peak / ON power (after)
- SRS blanking / SRS OFF / ON power RMS / ON power peak / ON power (after)

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.

6.2.1.13 Perror

class PerrorCls

Perror commands group definition. 15 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.13.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
```

class AverageCls

Average commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: Phase error value for low EVM window position
- High: List[float]: Phase error value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.perror.average.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
value: ResultData = driver.lteMeas.multiEval.perror.average.fetch()
```

Returns the values of the phase error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage
value: ResultData = driver.lteMeas.multiEval.perror.average.read()
```

Returns the values of the phase error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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.lteMeas.multiEval.perror.average.clone()
```

Subgroups

6.2.1.13.1.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available

- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage:NREF
value: ResultData = driver.lteMeas.multiEval.perror.average.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:AVERage:NREF
value: ResultData = driver.lteMeas.multiEval.perror.average.nref.read()
```

No command help available

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

6.2.1.13.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
```

class CurrentCls

Current commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: Phase error value for low EVM window position
- High: List[float]: Phase error value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.perror.current.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
value: ResultData = driver.lteMeas.multiEval.perror.current.fetch()
```

Returns the values of the phase error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent
value: ResultData = driver.lteMeas.multiEval.perror.current.read()
```

Returns the values of the phase error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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.lteMeas.multiEval.perror.current.clone()
```

Subgroups

6.2.1.13.2.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent:NREF
value: ResultData = driver.lteMeas.multiEval.perror.current.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:CURRent:NREF
value: ResultData = driver.lteMeas.multiEval.perror.current.nref.read()
```

No command help available

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

6.2.1.13.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 5 total commands, 1 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: Phase error value for low EVM window position
- High: List[float]: Phase error value for high EVM window position

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.perror.maximum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
value: ResultData = driver.lteMeas.multiEval.perror.maximum.fetch()
```

Returns the values of the phase error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum
value: ResultData = driver.lteMeas.multiEval.perror.maximum.read()
```

Returns the values of the phase error diagrams for the SC-FDMA symbols in the measured slot. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per SC-FDMA symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Square Magnitude Error, Phase Error'.

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.lteMeas.multiEval.perror.maximum.clone()
```

Subgroups

6.2.1.13.3.1 Nref

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum:NREF
FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum:NREF
```

class NrefCls

Nref commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Sym_1_L: float: No parameter help available
- Sym_1_H: float: No parameter help available
- Sym_2_L: float: No parameter help available
- Sym_2_H: float: No parameter help available
- Sym_3_L: float: No parameter help available
- Sym_3_H: float: No parameter help available
- Sym_5_L: float: No parameter help available
- Sym_5_H: float: No parameter help available
- Sym_6_L: float: No parameter help available
- Sym_6_H: float: No parameter help available
- Sym_7_L: float: No parameter help available
- Sym_7_H: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum:NREF
value: ResultData = driver.lteMeas.multiEval.perror.maximum.nref.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PERRor:MAXimum:NREF
value: ResultData = driver.lteMeas.multiEval.perror.maximum.nref.read()
```

No command help available

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

6.2.1.14 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 65 total commands, 9 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.14.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.

- Tx_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.pmonitor.average.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.average.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.average.read()
```

Returns the total TX power of all carriers.

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

6.2.1.14.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.pmonitor.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.pmonitor.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

class CcCls

Cc commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

Cloning the Group

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

Subgroups

6.2.1.14.2.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.average.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

6.2.1.14.2.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.current.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

6.2.1.14.2.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.maximum.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.maximum.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

6.2.1.14.2.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.minimum.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.minimum.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

6.2.1.14.2.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.standardDev.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr>:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.cc.standardDev.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the TX power of carrier CC<no>.

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

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

6.2.1.14.3 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.pmonitor.current.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.current.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.current.read()
```

Returns the total TX power of all carriers.

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

6.2.1.14.4 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.pmonitor.maximum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.maximum.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.maximum.read()
```

Returns the total TX power of all carriers.

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

6.2.1.14.5 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: CalculateStruct = driver.lteMeas.multiEval.pmonitor.minimum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.minimum.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.minimum.read()
```

Returns the total TX power of all carriers.

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

6.2.1.14.6 Pcc

class PccCls

Pcc commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.14.6.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.average.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.average.read()
```

No command help available

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

6.2.1.14.6.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.current.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.current.read()
```

No command help available

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

6.2.1.14.6.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.maximum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.maximum.read()
```

No command help available

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

6.2.1.14.6.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.minimum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.minimum.read()
```

No command help available

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

6.2.1.14.6.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.standardDev.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:PCC:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.pcc.standardDev.read()
```

No command help available

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

6.2.1.14.7 Scc

class SccCls

Scc commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.14.7.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.average.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.average.read()
```

No command help available

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

6.2.1.14.7.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available

- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.current.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.current.read()
```

No command help available

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

6.2.1.14.7.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.maximum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.maximum.read()
```

No command help available

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

6.2.1.14.7.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.minimum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.minimum.read()
```

No command help available

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

6.2.1.14.7.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.standardDev.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SCC:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.scc.standardDev.read()
```

No command help available

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

6.2.1.14.8 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power measurements exceeding the specified power limits.
- Tx_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.pmonitor.standardDev.
↪calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.standardDev.fetch()
```

Returns the total TX power of all carriers.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.standardDev.read()
```

Returns the total TX power of all carriers.

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

6.2.1.14.9 Ulca

class UlcaCls

Ulca commands group definition. 20 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.14.9.1 Pcc

class PccCls

Pcc commands group definition. 10 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.14.9.2 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.average.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.average.read()
```

No command help available

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

6.2.1.14.9.3 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.current.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.current.read()
```

No command help available

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

6.2.1.14.9.4 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.maximum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.maximum.read()
```

No command help available

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

6.2.1.14.9.5 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.minimum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.minimum.read()
```

No command help available

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

6.2.1.14.9.6 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.standardDev.
↪fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:PCC:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.pcc.standardDev.
↪read()
```

No command help available

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

6.2.1.14.9.7 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.lteMeas.multiEval.pmonitor.ulca.scc.repcap_secondaryCC_get()
driver.lteMeas.multiEval.pmonitor.ulca.scc.repcap_secondaryCC_set(repcap.SecondaryCC.CC1)
```

class SccCls

Scc commands group definition. 10 total commands, 5 Subgroups, 0 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

Cloning the Group

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

Subgroups

6.2.1.14.9.8 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>
↪:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.average.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

read(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:AVERage
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.average.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.14.9.9 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>
↪:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.current.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

read(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:CURRent
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.current.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.14.9.10 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>
↪:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.maximum.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

read(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:MAXimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.maximum.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.14.9.11 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>
↪:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.minimum.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

read(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:MINimum
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.minimum.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.14.9.12 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>
↪:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.standardDev.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

read(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:PMONitor:ULCA:SCC<Nr>
↪:SDEViation
value: ResultData = driver.lteMeas.multiEval.pmonitor.ulca.scc.standardDev.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

6.2.1.15 ReferenceMarker

class ReferenceMarkerCls

ReferenceMarker commands group definition. 6 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.15.1 EvMagnitude

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude
```

class EvMagnitudeCls

EvMagnitude commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude
value: float = driver.lteMeas.multiEval.referenceMarker.evMagnitude.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

return
yvalue: Absolute y-value of the marker position

Cloning the Group

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

Subgroups

6.2.1.15.1.1 Peak

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:EVMagnitude:PEAK
value: float = driver.lteMeas.multiEval.referenceMarker.evMagnitude.peak.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

return
yvalue: Absolute y-value of the marker position

6.2.1.15.2 Merror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:MERRor
```

class MerrorCls

Merror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:MERRor
value: float = driver.lteMeas.multiEval.referenceMarker.merror.fetch(xvalue = 1,
↪ trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

return
yvalue: Absolute y-value of the marker position

6.2.1.15.3 Pdynamics

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PDYNamics
```

class PdynamicsCls

Pdynamics commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: float*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PDYNamics
value: float = driver.lteMeas.multiEval.referenceMarker.pdynamics.fetch(xvalue
↪= 1.0, trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the power dynamics trace.

Suppressed linked return values: reliability

param xvalue
(float or boolean) Absolute x-value of the marker position

param trace_select
No help available

return
yvalue: Absolute y-value of the marker position

6.2.1.15.4 Perror

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PERRor
```

class PerrorCls

Perror commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*, *trace_select: TraceSelect*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PERRor
value: float = driver.lteMeas.multiEval.referenceMarker.perror.fetch(xvalue = 1,
↪ trace_select = enums.TraceSelect.AVERage)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs SC-FDMA symbol.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position There are two x-values per SC-FDMA symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 6 low, symbol 6 high) .

param trace_select
No help available

return
yvalue: Absolute y-value of the marker position

6.2.1.15.5 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.15.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.referenceMarker.pmonitor.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.referenceMarker.pmonitor.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*xvalue: int*, *carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:REFMarker:PMONitor:CC<Nr>
value: float = driver.lteMeas.multiEval.referenceMarker.pmonitor.cc.
↪fetch(xvalue = 1, carrierComponent = repcap.CarrierComponent.Default)
```

Uses the reference marker on the power monitor trace.

Suppressed linked return values: reliability

param xvalue
(integer or boolean) Absolute x-value of the marker position (subframe number)

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

return
yvalue: Absolute y-value of the marker position

Cloning the Group

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

6.2.1.16 SeMask

class SeMaskCls

SeMask commands group definition. 28 total commands, 9 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.16.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float or bool: Occupied bandwidth
- Tx_Power: float or bool: Total TX power in the slot over all component carriers

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot over all component carriers

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: CalculateStruct = driver.lteMeas.multiEval.seMask.average.calculate()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: ResultData = driver.lteMeas.multiEval.seMask.average.fetch()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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:LTE:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: ResultData = driver.lteMeas.multiEval.seMask.average.read()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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.

6.2.1.16.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float or bool: Occupied bandwidth
- Tx_Power: float or bool: Total TX power in the slot over all component carriers

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot over all component carriers

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: CalculateStruct = driver.lteMeas.multiEval.seMask.current.calculate()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: ResultData = driver.lteMeas.multiEval.seMask.current.fetch()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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:LTE:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: ResultData = driver.lteMeas.multiEval.seMask.current.read()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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.

6.2.1.16.3 Dallocation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DALLocation
```

class DallocationCls

Dallocation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Nr_Res_Blocks: int: Number of allocated resource blocks
- Offset_Res_Blocks: int: Offset of the first allocated resource block from the edge of the allocated UL transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DALLocation
value: FetchStruct = driver.lteMeas.multiEval.seMask.dallocation.fetch()
```

Returns the detected allocation for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

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

6.2.1.16.4 DchType

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → UplinkChannelType

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:DCHType
value: enums.UplinkChannelType = driver.lteMeas.multiEval.seMask.dchType.fetch()
```

Returns the uplink channel type for the measured slot. If the same slot is measured by the individual measurements, all commands yield the same result. If different statistic counts are defined for the modulation, ACLR and spectrum emission mask measurements, different slots can be measured and different results can be returned by the individual commands.

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.16.5 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
```

class ExtremeCls

Extreme commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float or bool: Occupied bandwidth
- Tx_Power_Min: float or bool: Minimum total TX power in the slot
- Tx_Power_Max: float or bool: Maximum total TX power in the slot

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float: Occupied bandwidth
- Tx_Power_Min: float: Minimum total TX power in the slot
- Tx_Power_Max: float: Maximum total TX power in the slot

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: CalculateStruct = driver.lteMeas.multiEval.seMask.extreme.calculate()
```

Return the extreme single-value results of the spectrum emission 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:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: ResultData = driver.lteMeas.multiEval.seMask.extreme.fetch()
```

Return the extreme single-value results of the spectrum emission 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:LTE:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: ResultData = driver.lteMeas.multiEval.seMask.extreme.read()
```

Return the extreme single-value results of the spectrum emission 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.

6.2.1.16.6 Margin

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin
```

class MarginCls

Margin commands group definition. 8 total commands, 4 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Margin_Curr_Neg: List[float]: No parameter help available
- Margin_Curr_Pos: List[float]: No parameter help available
- Margin_Avg_Neg: List[float]: No parameter help available
- Margin_Avg_Pos: List[float]: No parameter help available
- Margin_Min_Neg: List[float]: No parameter help available
- Margin_Min_Pos: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.fetch()
```

No command help available

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.lteMeas.multiEval.seMask.margin.clone()
```

Subgroups

6.2.1.16.6.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Margin_Curr_Neg: List[float]: No parameter help available
- Margin_Curr_Pos: List[float]: No parameter help available

- Margin_Avg_Neg: List[float]: No parameter help available
- Margin_Avg_Pos: List[float]: No parameter help available
- Margin_Min_Neg: List[float]: No parameter help available
- Margin_Min_Pos: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:ALL
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.all.fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. Results are provided for the current, average and maximum traces. For each trace, 24 values related to the negative (Neg) and positive (Pos) offset frequencies of emission mask areas 1 to 12 are provided. For inactive areas, NCAP is returned.

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

6.2.1.16.6.2 Average

class AverageCls

Average commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.16.6.3 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:AVERage:NEGativ
```

class NegativCls

Negativ commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Margin_Avg_Neg_X: List[float]: No parameter help available
- Margin_Avg_Neg_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:AVERage:NEGativ
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.average.negativ.
↪fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the x- and y-values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.1.16.6.4 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:AVERage:POSitiv
```

class PositivCls

Positiv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Margin_Avg_Pos_X: List[float]: No parameter help available
- Margin_Avg_Pos_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:AVERage:POSitiv
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.average.positiv.
↪fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the x- and y-values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.1.16.6.5 Current

class CurrentCls

Current commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.16.6.6 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:CURRent:NEGativ
```

class NegativCls

Negativ commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Margin_Curr_Neg_X: List[float]: No parameter help available
- Margin_Curr_Neg_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:CURRent:NEGativ
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.current.negativ.
↪fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the x- and y-values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

return

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

6.2.1.16.6.7 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:CURRent:POSitiv
```

class PositivCls

Positiv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Margin_Curr_Pos_X: List[float]: No parameter help available
- Margin_Curr_Pos_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:CURRent:POSitiv
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.current.positiv.
↪fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the x- and y-values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.1.16.6.8 Minimum

class MinimumCls

Minimum commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.16.6.9 Negativ

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:MINimum:NEGativ
```

class NegativCls

Negativ commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Margin_Min_Neg_X: List[float]: No parameter help available
- Margin_Min_Neg_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:MINimum:NEGativ
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.minimum.negativ.
↪fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the x- and y-values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.1.16.6.10 Positiv

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MARGin:MINimum:POSitiv
```

class PositivCls

Positiv commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Margin_Min_Pos_X: List[float]: No parameter help available
- Margin_Min_Pos_Y: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:MINimum:POSitiv
value: FetchStruct = driver.lteMeas.multiEval.seMask.margin.minimum.positiv.
↪fetch()
```

Returns spectrum emission mask margin results. A negative margin indicates that the trace is located above the limit line, i.e. the limit is exceeded. The individual commands provide results for the CURRent, AVERage and maximum traces (resulting in MINimum margins) . For each trace, the x- and y-values of the margins for emission mask areas 1 to 12 are provided for NEGative and POSitive offset frequencies. For inactive areas, NCAP is returned. Returned sequence: <Reliability>, <OutOfTolerance>, {<MarginX>, <MarginY>}area1, {...}area2, ..., {...}area12

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

6.2.1.16.7 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MAXimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Obw: float or bool: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Obw: float: No parameter help available
- Tx_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:MAXimum
value: CalculateStruct = driver.lteMeas.multiEval.seMask.maximum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MAXimum
value: ResultData = driver.lteMeas.multiEval.seMask.maximum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:SEMask:MAXimum
value: ResultData = driver.lteMeas.multiEval.seMask.maximum.read()
```

No command help available

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

6.2.1.16.8 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:MINimum
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MINimum
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Tx_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:MINimum
value: CalculateStruct = driver.lteMeas.multiEval.seMask.minimum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:MINimum
value: ResultData = driver.lteMeas.multiEval.seMask.minimum.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:SEMask:MINimum
value: ResultData = driver.lteMeas.multiEval.seMask.minimum.read()
```

No command help available

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

6.2.1.16.9 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Obw: float or bool: No parameter help available
- Tx_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot over all component carriers

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
value: CalculateStruct = driver.lteMeas.multiEval.seMask.standardDev.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
value: ResultData = driver.lteMeas.multiEval.seMask.standardDev.fetch()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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:LTE:MEASurement<Instance>:MEValuation:SEMask:SDEViation
value: ResultData = driver.lteMeas.multiEval.seMask.standardDev.read()
```

Return the current, average and standard deviation single-value results of the spectrum emission 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.

6.2.1.17 State

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:STATe
```

class StateCls

State commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch(*timeout: float = None*, *target_main_state: TargetStateA = None*, *target_sync_state: TargetSyncState = None*) → ResourceState

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:STATe
value: enums.ResourceState = driver.lteMeas.multiEval.state.fetch(timeout = 1.0,
↪ target_main_state = enums.TargetStateA.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state. Without query parameters, the state is returned immediately. With query parameters, the state is returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

param timeout
No help available

param target_main_state
Target MainState for the query Default is RUN.

param target_sync_state
Target SyncState for the query Default is ADJ.

return
meas_status: Current state or target state of ongoing state transition OFF: measurement off RUN: measurement running RDY: measurement completed

Cloning the Group

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

Subgroups

6.2.1.17.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:STATe:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*timeout: float = None*, *target_main_state: TargetStateA = None*, *target_sync_state: TargetSyncState = None*) → List[ResourceState]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:STATe:ALL
value: List[enums.ResourceState] = driver.lteMeas.multiEval.state.all.
↪fetch(timeout = 1.0, target_main_state = enums.TargetStateA.OFF, target_sync_
↪state = enums.TargetSyncState.ADJusted)
```

Queries the main measurement state and the measurement substates. Without query parameters, the states are returned immediately. With query parameters, the states are returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

param timeout
No help available

param target_main_state
Target MainState for the query Default is RUN.

param target_sync_state
Target SyncState for the query Default is ADJ.

return
state: No help available

6.2.1.18 Trace

class TraceCls

Trace commands group definition. 60 total commands, 10 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.1 Aclr

class AclrCls

Aclr commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Utra_2_Neg: float: Power in the second UTRA channel with lower frequency

- Utra_1_Neg: float: Power in the first UTRA channel with lower frequency
- Eutra_Negativ: float: Power in the first E-UTRA channel with lower frequency
- Eutra: float: Power in the allocated E-UTRA channel
- Eutra_Positiv: float: Power in the first E-UTRA channel with higher frequency
- Utra_1_Pos: float: Power in the first UTRA channel with higher frequency
- Utra_2_Pos: float: Power in the second UTRA channel with higher frequency

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
value: ResultData = driver.lteMeas.multiEval.trace.aclr.average.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
value: ResultData = driver.lteMeas.multiEval.trace.aclr.average.read()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'.

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

6.2.1.18.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Utra_2_Neg: float: Power in the second UTRA channel with lower frequency
- Utra_1_Neg: float: Power in the first UTRA channel with lower frequency
- Eutra_Negativ: float: Power in the first E-UTRA channel with lower frequency
- Eutra: float: Power in the allocated E-UTRA channel
- Eutra_Positiv: float: Power in the first E-UTRA channel with higher frequency
- Utra_1_Pos: float: Power in the first UTRA channel with higher frequency

- Utra_2_Pos: float: Power in the second UTRA channel with higher frequency

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
value: ResultData = driver.lteMeas.multiEval.trace.aclr.current.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
value: ResultData = driver.lteMeas.multiEval.trace.aclr.current.read()
```

Returns the absolute powers as displayed in the ACLR diagram. The current and average values can be retrieved. See also 'Square Spectrum ACLR'.

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

6.2.1.18.2 EsFlatness

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
```

class EsFlatnessCls

EsFlatness commands group definition. 4 total commands, 1 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
value: List[float] = driver.lteMeas.multiEval.trace.esFlatness.fetch()
```

Returns the values of the equalizer spectrum flatness trace. See also 'Square Equalizer Spectrum Flatness'.

Suppressed linked return values: reliability

return
power: Comma-separated list of power values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness
value: List[float] = driver.lteMeas.multiEval.trace.esFlatness.read()
```

Returns the values of the equalizer spectrum flatness trace. See also 'Square Equalizer Spectrum Flatness'.

Suppressed linked return values: reliability

return
power: Comma-separated list of power values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

Cloning the Group

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

Subgroups

6.2.1.18.2.1 Phase

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
```

class PhaseCls

Phase commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
value: List[float] = driver.lteMeas.multiEval.trace.esFlatness.phase.fetch()
```

Returns the phase values of the equalizer spectrum flatness trace. The GUI shows only the magnitude values.

Suppressed linked return values: reliability

return
phase: Comma-separated list of phase values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:ESFLatness:PHASe
value: List[float] = driver.lteMeas.multiEval.trace.esFlatness.phase.read()
```

Returns the phase values of the equalizer spectrum flatness trace. The GUI shows only the magnitude values.

Suppressed linked return values: reliability

return
phase: Comma-separated list of phase values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

6.2.1.18.3 Evmc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
```

class EvmcCls

Evmc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
value: List[float] = driver.lteMeas.multiEval.trace.evmc.fetch()
```

Returns the values of the EVM vs subcarrier trace. See also 'Square EVM vs Subcarrier'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMC
value: List[float] = driver.lteMeas.multiEval.trace.evmc.read()
```

Returns the values of the EVM vs subcarrier trace. See also 'Square EVM vs Subcarrier'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

6.2.1.18.4 EvmSymbol

class EvmSymbolCls

EvmSymbol commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
value: List[float] = driver.lteMeas.multiEval.trace.evmSymbol.average.fetch()
```

Returns the values of the EVM vs modulation symbol trace. See also 'Square EVM'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:AVERage
value: List[float] = driver.lteMeas.multiEval.trace.evmSymbol.average.read()
```

Returns the values of the EVM vs modulation symbol trace. See also 'Square EVM'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

6.2.1.18.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
value: List[float] = driver.lteMeas.multiEval.trace.evmSymbol.current.fetch()
```

Returns the values of the EVM vs modulation symbol trace. See also 'Square EVM'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:CURRent
value: List[float] = driver.lteMeas.multiEval.trace.evmSymbol.current.read()
```

Returns the values of the EVM vs modulation symbol trace. See also 'Square EVM'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

6.2.1.18.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
value: List[float] = driver.lteMeas.multiEval.trace.evmSymbol.maximum.fetch()
```

Returns the values of the EVM vs modulation symbol trace. See also 'Square EVM'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:EVMSymbol:MAXimum
value: List[float] = driver.lteMeas.multiEval.trace.evmSymbol.maximum.read()
```

Returns the values of the EVM vs modulation symbol trace. See also 'Square EVM'.

Suppressed linked return values: reliability

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

6.2.1.18.5 Iemissions

class IemissionsCls

Iemissions commands group definition. 10 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.5.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.trace.iemissions.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.trace.iemissions.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
```

class CcCls

Cc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.cc.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the values of the in-band emissions trace for carrier CC<no>. See also 'Square Inband Emissions'.

Suppressed linked return values: reliability

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

return
power: Comma-separated list of power values, one value per resource block

read(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:CC<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.cc.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the values of the in-band emissions trace for carrier CC<no>. See also 'Square Inband Emissions'.

Suppressed linked return values: reliability

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

return
power: Comma-separated list of power values, one value per resource block

Cloning the Group

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

6.2.1.18.5.2 Pcc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions[:PCC]
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions[:PCC]
```

class PccCls

Pcc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.pcc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.pcc.read()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

6.2.1.18.5.3 Scc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:SCC
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:SCC
```

class SccCls

Scc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:SCC
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.scc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:SCC
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.scc.read()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

6.2.1.18.5.4 Ulca

class UlcaCls

Ulca commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.5.5 Pcc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA[:PCC]
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA[:PCC]
```

class PccCls

Pcc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.ulca.pcc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.ulca.pcc.read()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

6.2.1.18.5.6 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.lteMeas.multiEval.trace.iemissions.ulca.scc.repcap_secondaryCC_get()
driver.lteMeas.multiEval.trace.iemissions.ulca.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA:SCC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA:SCC<Nr>
```

class SccCls

Scc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

fetch(*secondaryCC=SecondaryCC.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA:SCC
↪<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.ulca.scc.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

Suppressed linked return values: reliability

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
power: No help available

read(*secondaryCC=SecondaryCC.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:IEMissions:ULCA:SCC<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.iemissions.ulca.scc.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

Suppressed linked return values: reliability

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
power: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.trace.iemissions.ulca.scc.clone()
```

6.2.1.18.6 Iq

class IqCls

Iq commands group definition. 2 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.6.1 High

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:HIGH
```

class HighCls

High commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Iphase: List[float]: Normalized I amplitude
- Qphase: List[float]: Normalized Q amplitude

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:HIGH
value: FetchStruct = driver.lteMeas.multiEval.trace.iq.high.fetch()
```

Returns the results in the I/Q constellation diagram for low and high EVM window position. There is one pair of values per modulation symbol. The results are returned in the following order: <Reliability>, {<IPhase>, <QPhase>}symbol 1, . . . , {<IPhase>, <QPhase>}symbol n See also 'Square I/Q Constellation'

return

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

6.2.1.18.6.2 Low

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:LOW
```

class LowCls

Low commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Iphase: List[float]: Normalized I amplitude
- Qphase: List[float]: Normalized Q amplitude

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:IQ:LOW
value: FetchStruct = driver.lteMeas.multiEval.trace.iq.low.fetch()
```

Returns the results in the I/Q constellation diagram for low and high EVM window position. There is one pair of values per modulation symbol. The results are returned in the following order: <Reliability>, {<IPhase>, <QPhase>}symbol 1, ..., {<IPhase>, <QPhase>}symbol n See also 'Square I/Q Constellation'

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

6.2.1.18.7 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.7.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
value: List[float] = driver.lteMeas.multiEval.trace.pdynamics.average.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the Measure Subframe (0 µs) . The diagram in the GUI shows only a subsection of this trace.

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:AVERage
value: List[float] = driver.lteMeas.multiEval.trace.pdynamics.average.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the Measure Subframe (0 µs) . The diagram in the GUI shows only a subsection of this trace.

6.2.1.18.7.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
value: List[float] = driver.lteMeas.multiEval.trace.pdynamics.current.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the Measure Subframe (0 µs) . The diagram in the GUI shows only a subsection of this trace.

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:CURRent
value: List[float] = driver.lteMeas.multiEval.trace.pdynamics.current.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the Measure Subframe (0 µs) . The diagram in the GUI shows only a subsection of this trace.

6.2.1.18.7.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
value: List[float] = driver.lteMeas.multiEval.trace.pdynamics.maximum.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the Measure Subframe (0 µs) . The diagram in the GUI shows only a subsection of this trace.

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PDYNamics:MAXimum
value: List[float] = driver.lteMeas.multiEval.trace.pdynamics.maximum.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measure subframe. The values have a spacing of 1.5625 µs. The 705th value is at the start of the Measure Subframe (0 µs) . The diagram in the GUI shows only a subsection of this trace.

6.2.1.18.8 Pmonitor

class PmonitorCls

Pmonitor commands group definition. 10 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.8.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.trace.pmonitor.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.trace.pmonitor.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
```

class CcCls

Cc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.cc.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the power monitor results for all captured CC<no> subframes.

Suppressed linked return values: reliability

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

return
power: Comma-separated list of power values, one value per subframe

read(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:CC<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.cc.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the power monitor results for all captured CC<no> subframes.

Suppressed linked return values: reliability

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

return
power: Comma-separated list of power values, one value per subframe

Cloning the Group

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

6.2.1.18.8.2 Pcc

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor[:PCC]
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor[:PCC]
```

class PccCls

Pcc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.pcc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.pcc.read()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

6.2.1.18.8.3 Scc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SCC
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SCC
```

class SccCls

Scc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SCC
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.scc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SCC
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.scc.read()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

6.2.1.18.8.4 Ulca

class UlcaCls

Ulca commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.8.5 Pcc

SCPI Commands :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA[:PCC]
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA[:PCC]
```

class PccCls

Pcc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.ulca.pcc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA[:PCC]
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.ulca.pcc.read()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

6.2.1.18.8.6 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.lteMeas.multiEval.trace.pmonitor.ulca.scc.repcap_secondaryCC_get()
driver.lteMeas.multiEval.trace.pmonitor.ulca.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA:SCC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA:SCC<Nr>
```

class SccCls

Scc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

fetch(*secondaryCC=SecondaryCC.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA:SCC<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.ulca.scc.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

Suppressed linked return values: reliability

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
power: No help available

read(*secondaryCC=SecondaryCC.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:PMONitor:ULCA:SCC<Nr>
value: List[float] = driver.lteMeas.multiEval.trace.pmonitor.ulca.scc.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

Suppressed linked return values: reliability

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
power: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.multiEval.trace.pmonitor.ulca.scc.clone()
```

6.2.1.18.9 RbaTable

class RbaTableCls

RbaTable commands group definition. 10 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.9.1 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.lteMeas.multiEval.trace.rbaTable.cc.repcap_carrierComponent_get()
driver.lteMeas.multiEval.trace.rbaTable.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
```

class CcCls

Cc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: CarrierComponent, default value after init: CarrierComponent.Nr1

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Channel_Type: List[enums.RbTableChannelType]: PUSCh / PUCCH: for UL slot with RB allocation PSSCh / PSCCh / PSBCh: for SL subframe with RB allocation NONE: UL slot or SL subframe contains no allocated RBs. DL: DL slot (only for TDD UL measurements) SSUB: part of special SF (only for TDD UL measurements)
- Offset_Rb: List[int]: Offset of first allocated RB for the given channel type
- No_Rb: List[int]: Number of allocated RBs for the given channel type

fetch(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.cc.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the information of the CC<no> RB allocation table. See also 'Square RB Allocation Table'. For uplink measurements, there are three results per captured slot (n = number of captured subframes) : <Reliability>, {<ChannelType>, <OffsetRB>, <NoRB>}slot 1, ..., {...}slot (n*2) For sidelink measurements, there are six results per captured subframe (SF) , three for the PSCCH and three for the PSSCH: <Reliability>, {...}SF 1 (PSCCH) , {...}SF 1 (PSSCH) , ..., {...}SF n (PSCCH) , {.. .}SF n (PSSCH)

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

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

read(*carrierComponent=CarrierComponent.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:CC<Nr>
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.cc.
↪read(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the information of the CC<no> RB allocation table. See also 'Square RB Allocation Table'. For uplink measurements, there are three results per captured slot (n = number of captured subframes) : <Reliability>, {<ChannelType>, <OffsetRB>, <NoRB>}slot 1, ..., {...}slot (n*2) For sidelink measurements, there are six results per captured subframe (SF) , three for the PSCCH and three for the PSSCH: <Reliability>, {...}SF 1 (PSCCH) , {...}SF 1 (PSSCH) , ..., {...}SF n (PSCCH) , {.. .}SF n (PSSCH)

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

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.lteMeas.multiEval.trace.rbaTable.cc.clone()
```

6.2.1.18.9.2 Pcc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable[:PCC]
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable[:PCC]
```

class PccCls

Pcc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Channel_Type: List[enums.RbTableChannelType]: No parameter help available
- Offset_Rb: List[int]: No parameter help available
- No_Rb: List[int]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable[:PCC]
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.pcc.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable[:PCC]
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.pcc.read()
```

No command help available

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

6.2.1.18.9.3 Scc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:SCC
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:SCC
```

class SccCls

Scc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Channel_Type: List[enums.RbTableChannelType]: No parameter help available
- Offset_Rb: List[int]: No parameter help available
- No_Rb: List[int]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:SCC
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.scc.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:SCC
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.scc.read()
```

No command help available

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

6.2.1.18.9.4 Ulca

class UlcaCls

Ulca commands group definition. 4 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.9.5 Pcc

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA[:PCC]
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA[:PCC]
```

class PccCls

Pcc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Channel_Type: List[enums.RbTableChannelType]: No parameter help available
- Offset_Rb: List[int]: No parameter help available
- No_Rb: List[int]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA[:PCC]
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.ulca.pcc.fetch()
```

No command help available

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA[:PCC]
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.ulca.pcc.read()
```

No command help available

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

6.2.1.18.9.6 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.lteMeas.multiEval.trace.rbaTable.ulca.scc.repcap_secondaryCC_get()
driver.lteMeas.multiEval.trace.rbaTable.ulca.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA:SCC<Nr>
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA:SCC<Nr>
```

class SccCls

Scc commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Channel_Type: List[enums.RbTableChannelType]: No parameter help available
- Offset_Rb: List[int]: No parameter help available
- No_Rb: List[int]: No parameter help available

fetch(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA:SCC<Nr>
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.ulca.scc.
↪fetch(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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

read(*secondaryCC=SecondaryCC.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:RBATable:ULCA:SCC<Nr>
value: ResultData = driver.lteMeas.multiEval.trace.rbaTable.ulca.scc.
↪read(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

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.lteMeas.multiEval.trace.rbaTable.ulca.scc.clone()
```

6.2.1.18.10 SeMask

class SeMaskCls

SeMask commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.18.10.1 Rbw<RBWkHz>

RepCap Settings

```
# Range: Rbw30 .. Rbw1000
rc = driver.lteMeas.multiEval.trace.seMask.rbw.repcap_rBWkHz_get()
driver.lteMeas.multiEval.trace.seMask.rbw.repcap_rBWkHz_set(repcap.RBWkHz.Rbw30)
```

class RbwCls

Rbw commands group definition. 6 total commands, 3 Subgroups, 0 group commands Repeated Capability: RBWkHz, default value after init: RBWkHz.Rbw30

Cloning the Group

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

Subgroups

6.2.1.18.10.2 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:AVERage
value: List[float] = driver.lteMeas.multiEval.trace.seMask.rbw.average.
↪fetch(rBWkHz = repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'Square Spectrum Emission Mask'.

Suppressed linked return values: reliability

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between adjacent results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned.

read(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
value: List[float] = driver.lteMeas.multiEval.trace.seMask.rbw.average.
↪read(rBWkHz = repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'Square Spectrum Emission Mask'.

Suppressed linked return values: reliability

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between adjacent results depends on the resolution bandwidth, see table below. For RBW100 and greater,

results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned.

6.2.1.18.10.3 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:CURRent
value: List[float] = driver.lteMeas.multiEval.trace.seMask.rbw.current.
↪fetch(rBWkHz = repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'Square Spectrum Emission Mask'.

Suppressed linked return values: reliability

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between adjacent results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned.

read(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
value: List[float] = driver.lteMeas.multiEval.trace.seMask.rbw.current.
↪read(rBWkHz = repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'Square Spectrum Emission Mask'.

Suppressed linked return values: reliability

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between adjacent

results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned.

6.2.1.18.10.4 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:MAXimum
value: List[float] = driver.lteMeas.multiEval.trace.seMask.rbw.maximum.
↪fetch(rBWkHz = repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'Square Spectrum Emission Mask'.

Suppressed linked return values: reliability

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: Comma-separated list of power results The value in the middle of the result array corresponds to the center frequency. The test point separation between adjacent results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned.

read(*rBWkHz=RBWkHz.Default*) → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
value: List[float] = driver.lteMeas.multiEval.trace.seMask.rbw.maximum.
↪read(rBWkHz = repcap.RBWkHz.Default)
```

Returns the values of the spectrum emission traces. Separate traces are available for the individual resolution bandwidths (<kHz>) . The results of the current, average and maximum traces can be retrieved. See also 'Square Spectrum Emission Mask'.

Suppressed linked return values: reliability

param rBWkHz
optional repeated capability selector. Default value: Rbw30 (settable in the interface 'Rbw')

return
power: Comma-separated list of power results The value in the middle of the result

array corresponds to the center frequency. The test point separation between adjacent results depends on the resolution bandwidth, see table below. For RBW100 and greater, results are only available for frequencies with active limits using these RBWs. For other frequencies, INV is returned.

6.2.1.19 VfThroughput

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:MEValuation:VFTHroughput
```

class VfThroughputCls

VfThroughput commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:MEValuation:VFTHroughput
value: float = driver.lteMeas.multiEval.vfThroughput.fetch()
```

Queries the View Filter Throughput.

Suppressed linked return values: reliability

return
vf_throughput: No help available

6.2.2 Prach

SCPI Commands :

```
INITiate:LTE:MEASurement<Instance>:PRACh
STOP:LTE:MEASurement<Instance>:PRACh
ABORt:LTE:MEASurement<Instance>:PRACh
```

class PrachCls

Prach commands group definition. 85 total commands, 5 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:LTE:MEASurement<Instance>:PRACh
driver.lteMeas.prach.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.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

initiate(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:LTE:MEASurement<Instance>:PRACh
driver.lteMeas.prach.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.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

stop() → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:PRACh
driver.lteMeas.prach.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.
    - 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(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:PRACh
driver.lteMeas.prach.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the RUN state.
```

(continues on next page)

(continued from previous page)

```
    - 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 RsCMPX_LteMeas.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

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

Subgroups

6.2.2.1 EvmSymbol

class EvmSymbolCls

EvmSymbol commands group definition. 15 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.2.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
CALCulate:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available

- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position.
- High: List[float]: EVM value for high EVM window position.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: CalculateStruct = driver.lteMeas.prach.evmSymbol.average.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: ResultData = driver.lteMeas.prach.evmSymbol.average.fetch()
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: ResultData = driver.lteMeas.prach.evmSymbol.average.read()
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

6.2.2.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
CALCulate:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position.
- High: List[float]: EVM value for high EVM window position.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: CalculateStruct = driver.lteMeas.prach.evmSymbol.current.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: ResultData = driver.lteMeas.prach.evmSymbol.current.fetch()
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: ResultData = driver.lteMeas.prach.evmSymbol.current.read()
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

6.2.2.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
CALCulate:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Low: List[enums.ResultStatus2]: No parameter help available
- High: List[enums.ResultStatus2]: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position.
- High: List[float]: EVM value for high EVM window position.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: CalculateStruct = driver.lteMeas.prach.evmSymbol.maximum.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: ResultData = driver.lteMeas.prach.evmSymbol.maximum.fetch()
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: ResultData = driver.lteMeas.prach.evmSymbol.maximum.read()
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

6.2.2.1.4 Peak

class PeakCls

Peak commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.evmSymbol.peak.clone()
```

Subgroups

6.2.2.1.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position.
- High: List[float]: EVM value for high EVM window position.

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
value: ResultData = driver.lteMeas.prach.evmSymbol.peak.average.fetch()
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
value: ResultData = driver.lteMeas.prach.evmSymbol.peak.average.read()
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

6.2.2.1.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position.
- High: List[float]: EVM value for high EVM window position.

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
value: ResultData = driver.lteMeas.prach.evmSymbol.peak.current.fetch()
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
value: ResultData = driver.lteMeas.prach.evmSymbol.peak.current.read()
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

6.2.2.1.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Low: List[float]: EVM value for low EVM window position.
- High: List[float]: EVM value for high EVM window position.

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.lteMeas.prach.evmSymbol.peak.maximum.fetch()
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

read() → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.lteMeas.prach.evmSymbol.peak.maximum.read()
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured preamble. The results of the current, average and maximum diagrams can be retrieved. There is one pair of EVM values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>,

<High>}symbol 1 If the preamble contains only one symbol, NCAPs are returned for the remaining symbol. See also 'Square EVM vs Symbol'.

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

6.2.2.2 Modulation

class ModulationCls

Modulation commands group definition. 21 total commands, 9 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.2.2.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position.
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position.
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position.
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position.
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position.
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position.
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position.
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position.
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position.
- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position.

- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position.
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position.
- Frequency_Error: float or bool: Carrier frequency error.
- Timing_Error: float or bool: Transmit time error.
- Tx_Power: float or bool: UE RMS power.
- Peak_Power: float or bool: UE peak power.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float: EVM RMS value, low EVM window position.
- Evm_Rms_High: float: EVM RMS value, high EVM window position.
- Evm_Peak_Low: float: EVM peak value, low EVM window position.
- Evm_Peak_High: float: EVM peak value, high EVM window position.
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position.
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position.
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position.
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position.
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position.
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position.
- Frequency_Error: float: Carrier frequency error.
- Timing_Error: float: Transmit time error.
- Tx_Power: float: UE RMS power.
- Peak_Power: float: UE peak power.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
value: CalculateStruct = driver.lteMeas.prach.modulation.average.calculate()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
value: ResultData = driver.lteMeas.prach.modulation.average.fetch()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:PRACh:MODulation:AVERage
value: ResultData = driver.lteMeas.prach.modulation.average.read()
```

Return the current, average and standard deviation single-value results. 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.

6.2.2.2.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position.
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position.
- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position.
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position.
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position.
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position.
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position.
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position.
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position.

- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position.
- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position.
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position.
- Frequency_Error: float or bool: Carrier frequency error.
- Timing_Error: float or bool: Transmit time error.
- Tx_Power: float or bool: UE RMS power.
- Peak_Power: float or bool: UE peak power.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float: EVM RMS value, low EVM window position.
- Evm_Rms_High: float: EVM RMS value, high EVM window position.
- Evm_Peak_Low: float: EVM peak value, low EVM window position.
- Evm_Peak_High: float: EVM peak value, high EVM window position.
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position.
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position.
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position.
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position.
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position.
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position.
- Frequency_Error: float: Carrier frequency error.
- Timing_Error: float: Transmit time error.
- Tx_Power: float: UE RMS power.
- Peak_Power: float: UE peak power.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
value: CalculateStruct = driver.lteMeas.prach.modulation.current.calculate()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
value: ResultData = driver.lteMeas.prach.modulation.current.fetch()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:PRACh:MODulation:CURRent
value: ResultData = driver.lteMeas.prach.modulation.current.read()
```

Return the current, average and standard deviation single-value results. 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.

6.2.2.2.3 DpfOffset

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset
```

class DpfOffsetCls

DpfOffset commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch() → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset
value: int = driver.lteMeas.prach.modulation.dpfOffset.fetch()
```

Returns the automatically detected or manually configured PRACH frequency offset for single-preamble measurements.

Suppressed linked return values: reliability

return
prach_freq_offset: PRACH frequency offset

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.modulation.dpfOffset.clone()
```

Subgroups

6.2.2.2.3.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.lteMeas.prach.modulation.dpfOffset.preamble.repcap_preamble_get()
driver.lteMeas.prach.modulation.dpfOffset.preamble.repcap_preamble_set(repcap.Preamble.
↪Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

fetch(*preamble=Preamble.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DPFoffset:PREamble
↪<Number>
value: int = driver.lteMeas.prach.modulation.dpfOffset.preamble.fetch(preamble␣
↪= repcap.Preamble.Default)
```

Returns the automatically detected or manually configured PRACH frequency offset for a selected preamble of multi-preamble measurements.

Suppressed linked return values: reliability

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

return
prach_freq_offset: PRACH frequency offset

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.modulation.dpfOffset.preamble.clone()
```

6.2.2.2.4 DsIndex

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex
```

class DsIndexCls

DsIndex commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch() → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex
value: int = driver.lteMeas.prach.modulation.dsIndex.fetch()
```

Returns the automatically detected or manually configured sequence index for single-preamble measurements.

Suppressed linked return values: reliability

return
sequence_index: Sequence index

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.modulation.dsIndex.clone()
```

Subgroups

6.2.2.2.4.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.lteMeas.prach.modulation.dsIndex.preamble.repcap_preamble_get()
driver.lteMeas.prach.modulation.dsIndex.preamble.repcap_preamble_set(repcap.Preamble.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

fetch(*preamble=Preamble.Default*) → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:DSINdex:PREamble
↪<Number>
value: int = driver.lteMeas.prach.modulation.dsIndex.preamble.fetch(preamble =␣
↪repcap.Preamble.Default)
```

Returns the automatically detected or manually configured sequence index for a selected preamble of multi-preamble measurements.

Suppressed linked return values: reliability

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

return
sequence_index: Sequence index

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.modulation.dsIndex.preamble.clone()
```

6.2.2.2.5 Extreme

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
```

class ExtremeCls

Extreme commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float or bool: EVM RMS value, low EVM window position
- Evm_Rms_High: float or bool: EVM RMS value, high EVM window position

- Evm_Peak_Low: float or bool: EVM peak value, low EVM window position
- Evm_Peak_High: float or bool: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float or bool: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float or bool: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float or bool: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float or bool: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float or bool: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float or bool: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float or bool: Phase error peak value, high EVM window position
- Frequency_Error: float or bool: Carrier frequency error
- Timing_Error: float or bool: Time error
- Tx_Power_Minimum: float or bool: Minimum UE RMS power
- Tx_Power_Maximum: float or bool: Maximum UE RMS power
- Peak_Power_Min: float or bool: Minimum UE peak power
- Peak_Power_Max: float or bool: Maximum UE peak power

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.
- Evm_Rms_Low: float: EVM RMS value, low EVM window position
- Evm_Rms_High: float: EVM RMS value, high EVM window position
- Evm_Peak_Low: float: EVM peak value, low EVM window position
- Evm_Peak_High: float: EVM peak value, high EVM window position
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position
- Frequency_Error: float: Carrier frequency error
- Timing_Error: float: Time error
- Tx_Power_Minimum: float: Minimum UE RMS power
- Tx_Power_Maximum: float: Maximum UE RMS power

- Peak_Power_Min: float: Minimum UE peak power
- Peak_Power_Max: float: Maximum UE peak power

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: CalculateStruct = driver.lteMeas.prach.modulation.extreme.calculate()
```

Returns the extreme single value results. 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:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: ResultData = driver.lteMeas.prach.modulation.extreme.fetch()
```

Returns the extreme single value results. 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:LTE:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: ResultData = driver.lteMeas.prach.modulation.extreme.read()
```

Returns the extreme single value results. 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.

6.2.2.2.6 Nsymbol

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:NSYMbol
```

class NsymbolCls

Nsymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → int

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:NSYMbol
value: int = driver.lteMeas.prach.modulation.nsymbol.fetch()
```

Queries the number of active OFDM symbols (symbols with result bars) in the EVM vs symbol diagram.

Suppressed linked return values: reliability

return
no_of_symbols: No help available

6.2.2.2.7 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.lteMeas.prach.modulation.preamble.repcap_preamble_get()
driver.lteMeas.prach.modulation.preamble.repcap_preamble_set(repcap.Preamble.Nr1)
```

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Preamble_Rel: int: Reliability indicator for the preamble.
- Evm_Rms_Low: float: EVM RMS value, low EVM window position.
- Evm_Rms_High: float: EVM RMS value, high EVM window position.
- Evm_Peak_Low: float: EVM peak value, low EVM window position.
- Evm_Peak_High: float: EVM peak value, high EVM window position.
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position.
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position.
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position.
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position.
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position.
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position.
- Frequency_Error: float: Carrier frequency error.
- Timing_Error: float: Transmit time error.
- Tx_Power: float: UE RMS power.
- Peak_Power: float: UE peak power.

fetch(*preamble=Preamble.Default*) → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
value: ResultData = driver.lteMeas.prach.modulation.preamble.fetch(preamble =␣
↪repcap.Preamble.Default)
```

Return the single value results of the EVM vs Preamble and Power vs Preamble views, for a selected preamble. See also 'Square EVM vs Preamble, Power vs Preamble'.

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

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

read(*preamble=Preamble.Default*) → ResultData

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
value: ResultData = driver.lteMeas.prach.modulation.preamble.read(preamble =␣
↪repcap.Preamble.Default)
```

Return the single value results of the EVM vs Preamble and Power vs Preamble views, for a selected preamble. See also 'Square EVM vs Preamble, Power vs Preamble'.

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

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.lteMeas.prach.modulation.preamble.clone()
```

6.2.2.2.8 Scorrelation

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation
```

class ScorrelationCls

Scorrelation commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch() → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation
value: float = driver.lteMeas.prach.modulation.scorrelation.fetch()
```

Returns the sequence correlation for single-preamble measurements. It indicates the correlation between the ideal preamble sequence determined from the parameter settings and the measured preamble sequence. A value of 1 corresponds to perfect correlation. A value close to 0 indicates that the preamble sequence was not found.

Suppressed linked return values: reliability

return
seq_correlation: Sequence correlation

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.modulation.scorrelation.clone()
```

Subgroups

6.2.2.2.8.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr400
rc = driver.lteMeas.prach.modulation.scorrelation.preamble.repcap_preamble_get()
driver.lteMeas.prach.modulation.scorrelation.preamble.repcap_preamble_set(repcap.
↪Preamble.Nr1)
```

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation:PREamble<Number>
```

class PreambleCls

Preamble commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: Preamble, default value after init: Preamble.Nr1

fetch(*preamble=Preamble.Default*) → float

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SCORrelation:PREamble
↪<Number>
value: float = driver.lteMeas.prach.modulation.scorrelation.preamble.
↪fetch(preamble = repcap.Preamble.Default)
```

Returns the sequence correlation for a selected preamble of multi-preamble measurements. It indicates the correlation between the ideal preamble sequence determined from the parameter settings and the measured preamble sequence. A value of 1 corresponds to perfect correlation. A value close to 0 indicates that the preamble sequence was not found.

Suppressed linked return values: reliability

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

return
seq_correlation: Sequence correlation

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.modulation.scorrelation.preamble.clone()
```

6.2.2.2.9 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Evm_Rms_Low: float or bool: No parameter help available
- Evm_Rms_High: float or bool: No parameter help available
- Evm_Peak_Low: float or bool: No parameter help available
- Evm_Peak_High: float or bool: No parameter help available
- Mag_Error_Rms_Low: float or bool: No parameter help available
- Mag_Error_Rms_High: float or bool: No parameter help available
- Mag_Error_Peak_Low: float or bool: No parameter help available
- Mag_Err_Peak_High: float or bool: No parameter help available
- Ph_Error_Rms_Low: float or bool: No parameter help available
- Ph_Error_Rms_High: float or bool: No parameter help available
- Ph_Error_Peak_Low: float or bool: No parameter help available
- Ph_Error_Peak_High: float or bool: No parameter help available
- Frequency_Error: float or bool: No parameter help available
- Timing_Error: float or bool: No parameter help available
- Tx_Power: float or bool: No parameter help available
- Peak_Power: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for modulation measurements exceeding the specified modulation limits.

- Evm_Rms_Low: float: EVM RMS value, low EVM window position.
- Evm_Rms_High: float: EVM RMS value, high EVM window position.
- Evm_Peak_Low: float: EVM peak value, low EVM window position.
- Evm_Peak_High: float: EVM peak value, high EVM window position.
- Mag_Error_Rms_Low: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Rms_High: float: Magnitude error RMS value, low EVM window position.
- Mag_Error_Peak_Low: float: Magnitude error peak value, low EVM window position.
- Mag_Err_Peak_High: float: Magnitude error peak value, high EVM window position.
- Ph_Error_Rms_Low: float: Phase error RMS value, low EVM window position.
- Ph_Error_Rms_High: float: Phase error RMS value, high EVM window position.
- Ph_Error_Peak_Low: float: Phase error peak value, low EVM window position.
- Ph_Error_Peak_High: float: Phase error peak value, high EVM window position.
- Frequency_Error: float: Carrier frequency error.
- Timing_Error: float: Transmit time error.
- Tx_Power: float: UE RMS power.
- Peak_Power: float: UE peak power.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
value: CalculateStruct = driver.lteMeas.prach.modulation.standardDev.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
value: ResultData = driver.lteMeas.prach.modulation.standardDev.fetch()
```

Return the current, average and standard deviation single-value results. 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:LTE:MEASurement<Instance>:PRACh:MODulation:SDEViation
value: ResultData = driver.lteMeas.prach.modulation.standardDev.read()
```

Return the current, average and standard deviation single-value results. 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.

6.2.2.3 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 15 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.2.3.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float or bool: ON power mean value over preamble.
- On_Power_Peak: float or bool: ON power peak value within preamble.
- Off_Power_After: float or bool: OFF power mean value for subframe after preamble without transient period.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for subframe before preamble without transient period.

- On_Power_Rms: float: ON power mean value over preamble.
- On_Power_Peak: float: ON power peak value within preamble.
- Off_Power_After: float: OFF power mean value for subframe after preamble without transient period.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: CalculateStruct = driver.lteMeas.prach.pdynamics.average.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: ResultData = driver.lteMeas.prach.pdynamics.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: ResultData = driver.lteMeas.prach.pdynamics.average.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.2.3.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float or bool: ON power mean value over preamble.
- On_Power_Peak: float or bool: ON power peak value within preamble.
- Off_Power_After: float or bool: OFF power mean value for subframe after preamble without transient period.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float: ON power mean value over preamble.
- On_Power_Peak: float: ON power peak value within preamble.
- Off_Power_After: float: OFF power mean value for subframe after preamble without transient period.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: CalculateStruct = driver.lteMeas.prach.pdynamics.current.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: ResultData = driver.lteMeas.prach.pdynamics.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: ResultData = driver.lteMeas.prach.pdynamics.current.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.2.3.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float or bool: ON power mean value over preamble.
- On_Power_Peak: float or bool: ON power peak value within preamble.
- Off_Power_After: float or bool: OFF power mean value for subframe after preamble without transient period.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float: ON power mean value over preamble.
- On_Power_Peak: float: ON power peak value within preamble.
- Off_Power_After: float: OFF power mean value for subframe after preamble without transient period.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: CalculateStruct = driver.lteMeas.prach.pdynamics.maximum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: ResultData = driver.lteMeas.prach.pdynamics.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: ResultData = driver.lteMeas.prach.pdynamics.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.2.3.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float or bool: ON power mean value over preamble.
- On_Power_Peak: float or bool: ON power peak value within preamble.
- Off_Power_After: float or bool: OFF power mean value for subframe after preamble without transient period.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'

- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float: ON power mean value over preamble.
- On_Power_Peak: float: ON power peak value within preamble.
- Off_Power_After: float: OFF power mean value for subframe after preamble without transient period.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: CalculateStruct = driver.lteMeas.prach.pdynamics.minimum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: ResultData = driver.lteMeas.prach.pdynamics.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: ResultData = driver.lteMeas.prach.pdynamics.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.2.3.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms: float or bool: No parameter help available
- On_Power_Peak: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for subframe before preamble without transient period.
- On_Power_Rms: float: ON power mean value over preamble.
- On_Power_Peak: float: ON power peak value within preamble.
- Off_Power_After: float: OFF power mean value for subframe after preamble without transient period.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
value: CalculateStruct = driver.lteMeas.prach.pdynamics.standardDev.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
value: ResultData = driver.lteMeas.prach.pdynamics.standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
value: ResultData = driver.lteMeas.prach.pdynamics.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.2.4 State

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:STATe
```

class StateCls

State commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch(*timeout: float = None*, *target_main_state: TargetStateA = None*, *target_sync_state: TargetSyncState = None*) → ResourceState

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:STATe
value: enums.ResourceState = driver.lteMeas.prach.state.fetch(timeout = 1.0,␣
↪target_main_state = enums.TargetStateA.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state. Without query parameters, the state is returned immediately. With query parameters, the state is returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

param timeout
No help available

param target_main_state
Target MainState for the query Default is RUN.

param target_sync_state
Target SyncState for the query Default is ADJ.

return
meas_status: Current state or target state of ongoing state transition OFF: measurement off RUN: measurement running RDY: measurement completed

Cloning the Group

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

Subgroups

6.2.2.4.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:STATe:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*timeout: float = None*, *target_main_state: TargetStateA = None*, *target_sync_state: TargetSyncState = None*) → List[ResourceState]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:STATe:ALL
value: List[enums.ResourceState] = driver.lteMeas.prach.state.all.fetch(timeout␣
↪= 1.0, target_main_state = enums.TargetStateA.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state and the measurement substates. Without query parameters, the states are returned immediately. With query parameters, the states are returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

param timeout
No help available

param target_main_state
Target MainState for the query Default is RUN.

param target_sync_state
Target SyncState for the query Default is ADJ.

return
state: No help available

6.2.2.5 Trace

class TraceCls

Trace commands group definition. 29 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.2.5.1 Evm

class EvmCls

Evm commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.trace.evm.clone()
```

Subgroups

6.2.2.5.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
value: List[float] = driver.lteMeas.prach.trace.evm.average.fetch()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
value: List[float] = driver.lteMeas.prach.trace.evm.average.read()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
value: List[float] = driver.lteMeas.prach.trace.evm.current.fetch()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
value: List[float] = driver.lteMeas.prach.trace.evm.current.read()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
value: List[float] = driver.lteMeas.prach.trace.evm.maximum.fetch()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
value: List[float] = driver.lteMeas.prach.trace.evm.maximum.read()
```

Return the values of the EVM vs subcarrier traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.2 EvPreamble

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
```

class EvPreambleCls

EvPreamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
value: List[float] = driver.lteMeas.prach.trace.evPreamble.fetch()
```

Return the values of the EVM vs preamble traces. See also 'Square EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: 32 EVM values, for preamble 1 to 32 (NCAP for not measured preambles)

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:EVPReamble
value: List[float] = driver.lteMeas.prach.trace.evPreamble.read()
```

Return the values of the EVM vs preamble traces. See also 'Square EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: 32 EVM values, for preamble 1 to 32 (NCAP for not measured preambles)

6.2.2.5.3 Iq

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:IQ
```

class IqCls

Iq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Iphase: List[float]: Normalized I amplitude
- Qphase: List[float]: Normalized Q amplitude

fetch() → FetchStruct

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:IQ
value: FetchStruct = driver.lteMeas.prach.trace.iq.fetch()
```

Returns the results in the I/Q constellation diagram. There is one pair of values per modulation symbol. For preamble format 4, there are 139 symbols. For preamble format 0 to 3, there are 839 symbols. The results are returned in the following order: <Reliability>, {<IPhase>, <QPhase>}symbol 1, ..., {<IPhase>, <QPhase>}symbol n See also 'Square I/Q Constellation'.

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

6.2.2.5.4 Merror

class MerrorCls

Merror commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.trace.merror.clone()
```

Subgroups

6.2.2.5.4.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
value: List[float] = driver.lteMeas.prach.trace.merror.average.fetch()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
value: List[float] = driver.lteMeas.prach.trace.merror.average.read()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.4.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
value: List[float] = driver.lteMeas.prach.trace.merror.current.fetch()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
value: List[float] = driver.lteMeas.prach.trace.merror.current.read()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.4.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
value: List[float] = driver.lteMeas.prach.trace.merror.maximum.fetch()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
value: List[float] = driver.lteMeas.prach.trace.merror.maximum.read()
```

Return the values of the magnitude error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.5 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.trace.pdynamics.clone()
```

Subgroups

6.2.2.5.5.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
value: List[float] = driver.lteMeas.prach.trace.pdynamics.average.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) .

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
value: List[float] = driver.lteMeas.prach.trace.pdynamics.average.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) .

6.2.2.5.5.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
value: List[float] = driver.lteMeas.prach.trace.pdynamics.current.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) .

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
value: List[float] = driver.lteMeas.prach.trace.pdynamics.current.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) .

6.2.2.5.5.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
value: List[float] = driver.lteMeas.prach.trace.pdynamics.maximum.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) .

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
value: List[float] = driver.lteMeas.prach.trace.pdynamics.maximum.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the preamble. The values have a spacing of 1.5625 µs. The 705th value is at the start of the preamble (0 µs) .

6.2.2.5.6 Perror

class PerrorCls

Perror commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.prach.trace.perror.clone()
```

Subgroups

6.2.2.5.6.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
value: List[float] = driver.lteMeas.prach.trace.perror.average.fetch()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
value: List[float] = driver.lteMeas.prach.trace.perror.average.read()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.6.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
value: List[float] = driver.lteMeas.prach.trace.perror.current.fetch()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
value: List[float] = driver.lteMeas.prach.trace.perror.current.read()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.6.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
value: List[float] = driver.lteMeas.prach.trace.perror.maximum.fetch()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
value: List[float] = driver.lteMeas.prach.trace.perror.maximum.read()
```

Return the values of the phase error traces. Each value is averaged over the samples in one preamble subcarrier. The results of the current, average and maximum traces can be retrieved. See also 'Square EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: The number of results depends on the preamble format. Format 0 to 3: 839 EVM values, format 4: 139 EVM values

6.2.2.5.7 PvPreamble

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
```

class PvPreambleCls

PvPreamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
value: List[float] = driver.lteMeas.prach.trace.pvPreamble.fetch()
```

Return the values of the power vs preamble traces. See also 'Square EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: 32 power values, for preamble 1 to 32 (NCAP for not measured preambles)

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:PRACh:TRACe:PVPReamble
value: List[float] = driver.lteMeas.prach.trace.pvPreamble.read()
```

Return the values of the power vs preamble traces. See also 'Square EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: 32 power values, for preamble 1 to 32 (NCAP for not measured preambles)

6.2.3 Srs

SCPI Commands :

```
INITiate:LTE:MEASurement<Instance>:SRS
STOP:LTE:MEASurement<Instance>:SRS
ABORt:LTE:MEASurement<Instance>:SRS
```

class SrsCls

Srs commands group definition. 26 total commands, 3 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:LTE:MEASurement<Instance>:SRS
driver.lteMeas.srs.abort()


    INTRO_CMD_HELP: Starts, stops or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
```

(continues on next page)

(continued from previous page)

```
↪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.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

initiate(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:LTE:MEASurement<Instance>:SRS
driver.lteMeas.srs.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.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

stop() → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:SRS
driver.lteMeas.srs.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.
    - 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(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:LTE:MEASurement<Instance>:SRS
driver.lteMeas.srs.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 RsCMPX_LteMeas.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

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

Subgroups

6.2.3.1 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 15 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.3.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float or bool: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float or bool: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float or bool: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float or bool: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float or bool: OFF power mean value for the subframe after the SRS symbol.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float: OFF power mean value for the subframe after the SRS symbol.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: CalculateStruct = driver.lteMeas.srs.pdynamics.average.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: ResultData = driver.lteMeas.srs.pdynamics.average.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: ResultData = driver.lteMeas.srs.pdynamics.average.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.3.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float or bool: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float or bool: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float or bool: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float or bool: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float or bool: OFF power mean value for the subframe after the SRS symbol.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float: OFF power mean value for the subframe after the SRS symbol.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: CalculateStruct = driver.lteMeas.srs.pdynamics.current.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: ResultData = driver.lteMeas.srs.pdynamics.current.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: ResultData = driver.lteMeas.srs.pdynamics.current.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.3.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float or bool: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float or bool: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float or bool: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float or bool: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float or bool: OFF power mean value for the subframe after the SRS symbol.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float: OFF power mean value for the subframe after the SRS symbol.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: CalculateStruct = driver.lteMeas.srs.pdynamics.maximum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: ResultData = driver.lteMeas.srs.pdynamics.maximum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: ResultData = driver.lteMeas.srs.pdynamics.maximum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.3.1.4 Minimum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
```

class MinimumCls

Minimum commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float or bool: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float or bool: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float or bool: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float or bool: ON power mean value over the second SRS symbol (NCAP returned for FDD) .

- On_Power_Peak_2: float or bool: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float or bool: OFF power mean value for the subframe after the SRS symbol.

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float: OFF power mean value for the subframe after the SRS symbol.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: CalculateStruct = driver.lteMeas.srs.pdynamics.minimum.calculate()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: ResultData = driver.lteMeas.srs.pdynamics.minimum.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: ResultData = driver.lteMeas.srs.pdynamics.minimum.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.3.1.5 StandardDev

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 3 total commands, 0 Subgroups, 3 group commands

class CalculateStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Out_Of_Tolerance: int: No parameter help available
- Off_Power_Before: float or bool: No parameter help available
- On_Power_Rms_1: float or bool: No parameter help available
- On_Power_Peak_1: float or bool: No parameter help available
- On_Power_Rms_2: float or bool: No parameter help available
- On_Power_Peak_2: float or bool: No parameter help available
- Off_Power_After: float or bool: No parameter help available

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Out_Of_Tolerance: int: Out of tolerance result, i.e. the percentage of measurement intervals of the statistic count for power dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power mean value for the time period before the SRS symbol.
- On_Power_Rms_1: float: ON power mean value over the first SRS symbol.
- On_Power_Peak_1: float: ON power peak value for the first SRS symbol.
- On_Power_Rms_2: float: ON power mean value over the second SRS symbol (NCAP returned for FDD) .
- On_Power_Peak_2: float: ON power peak value for the second SRS symbol (NCAP returned for FDD) .
- Off_Power_After: float: OFF power mean value for the subframe after the SRS symbol.

calculate() → CalculateStruct

```
# SCPI: CALCulate:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: CalculateStruct = driver.lteMeas.srs.pdynamics.standardDev.calculate()
```

No command help available

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

fetch() → ResultData

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: ResultData = driver.lteMeas.srs.pdynamics.standardDev.fetch()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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:LTE:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: ResultData = driver.lteMeas.srs.pdynamics.standardDev.read()
```

Return the current, average, minimum, maximum and standard deviation single-value results of the power dynamics 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.

6.2.3.2 State

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:SRS:STATe
```

class StateCls

State commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch(*timeout: float = None*, *target_main_state: TargetStateA = None*, *target_sync_state: TargetSyncState = None*) → ResourceState

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:STATe
value: enums.ResourceState = driver.lteMeas.srs.state.fetch(timeout = 1.0,␣
↪target_main_state = enums.TargetStateA.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state. Without query parameters, the state is returned immediately. With query parameters, the state is returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

param timeout
No help available

param target_main_state
Target MainState for the query Default is RUN.

param target_sync_state
Target SyncState for the query Default is ADJ.

return
meas_status: Current state or target state of ongoing state transition OFF: measurement off RUN: measurement running RDY: measurement completed

Cloning the Group

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

Subgroups

6.2.3.2.1 All

SCPI Command :

```
FETCh:LTE:MEASurement<Instance>:SRS:STATe:ALL
```

class AllCls

All commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*timeout: float = None*, *target_main_state: TargetStateA = None*, *target_sync_state: TargetSyncState = None*) → List[ResourceState]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:STATe:ALL
value: List[enums.ResourceState] = driver.lteMeas.srs.state.all.fetch(timeout =␣
↪1.0, target_main_state = enums.TargetStateA.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

Queries the main measurement state and the measurement substates. Without query parameters, the states are returned immediately. With query parameters, the states are returned when the <TargetMainState> and the <TargetSyncState> are reached or when the <Timeout> expires.

param timeout
No help available

param target_main_state
Target MainState for the query Default is RUN.

param target_sync_state
Target SyncState for the query Default is ADJ.

return
state: No help available

6.2.3.3 Trace

class TraceCls

Trace commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.3.3.1 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 6 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.lteMeas.srs.trace.pdynamics.clone()
```

Subgroups

6.2.3.3.1.1 Average

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
value: List[float] = driver.lteMeas.srs.trace.pdynamics.average.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via a remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) .

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
value: List[float] = driver.lteMeas.srs.trace.pdynamics.average.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via a remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) .

6.2.3.3.1.2 Current

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

`fetch()` → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
value: List[float] = driver.lteMeas.srs.trace.pdynamics.current.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via a remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) .

`read()` → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
value: List[float] = driver.lteMeas.srs.trace.pdynamics.current.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via a remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) .

6.2.3.3.1.3 Maximum

SCPI Commands :

```
READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
value: List[float] = driver.lteMeas.srs.trace.pdynamics.maximum.fetch()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via a remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) .

read() → List[float]

```
# SCPI: READ:LTE:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
value: List[float] = driver.lteMeas.srs.trace.pdynamics.maximum.read()
```

Return the values of the power dynamics traces. Each value is sampled with 48 Ts, corresponding to 1.5625 µs. The results of the current, average and maximum traces can be retrieved. Note that the GUI shows only the beginning of the trace returned via a remote command. The last 800 µs cannot be displayed at the GUI. See also 'Measurement results'.

Suppressed linked return values: reliability

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the SRS symbol. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS symbol (0 µs) .

6.3 Sense

class SenseCls

Sense commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1 LteMeas

class LteMeasCls

LteMeas commands group definition. 6 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1.1 CarrierAggregation

SCPI Command :

```
SENSe:LTE:MEASurement<Instance>:CAGGregation:FSHWare
```

class CarrierAggregationCls

CarrierAggregation commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_fshware() → bool

```
# SCPI: SENSe:LTE:MEASurement<Instance>:CAGGregation:FSHWare
value: bool = driver.sense.lteMeas.carrierAggregation.get_fshware()
```

No command help available

return
value: No help available

6.3.1.2 MultiEval

class MultiEvalCls

MultiEval commands group definition. 5 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1.2.1 Limit

class LimitCls

Limit commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1.2.1.1 Iemissions

SCPI Commands :

```
SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions:SCC
SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions[:PCC]
```

class IemissionsCls

Iemissions commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_pcc() → List[float]

```
# SCPI: SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions[:PCC]
value: List[float] = driver.sense.lteMeas.multiEval.limit.iemissions.get_pcc()
```

No command help available

return
power: No help available

get_scc() → List[float]

```
# SCPI: SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions:SCC
value: List[float] = driver.sense.lteMeas.multiEval.limit.iemissions.get_scc()
```

No command help available

return
power: No help available

Cloning the Group

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

Subgroups

6.3.1.2.1.2 Ulca

SCPI Command :

```
SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions:ULCA[:PCC]
```

class UlcaCls

Ulca commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_pcc() → List[float]

```
# SCPI: SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions:ULCA[:PCC]
value: List[float] = driver.sense.lteMeas.multiEval.limit.iemissions.ulca.get_
↪pcc()
```

No command help available

return
power: No help available

Cloning the Group

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

Subgroups

6.3.1.2.1.3 Scc<SecondaryCC>

RepCap Settings

```
# Range: CC1 .. CC7
rc = driver.sense.lteMeas.multiEval.limit.iemissions.ulca.scc.repcap_secondaryCC_get()
driver.sense.lteMeas.multiEval.limit.iemissions.ulca.scc.repcap_secondaryCC_set(repcap.
↪SecondaryCC.CC1)
```

SCPI Command :

```
SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions:ULCA:SCC<Nr>
```

class SccCls

Scc commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: SecondaryCC, default value after init: SecondaryCC.CC1

get(*secondaryCC*=*SecondaryCC.Default*) → List[float]

```
# SCPI: SENSe:LTE:MEASurement<Instance>:MEValuation:LIMit:IEMissions:ULCA:SCC
↪<Nr>
value: List[float] = driver.sense.lteMeas.multiEval.limit.iemissions.ulca.scc.
↪get(secondaryCC = repcap.SecondaryCC.Default)
```

No command help available

param secondaryCC
optional repeated capability selector. Default value: CC1 (settable in the interface 'Scc')

return
power: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.lteMeas.multiEval.limit.iemissions.ulca.scc.clone()
```

6.3.1.2.2 Spectrum

class SpectrumCls

Spectrum commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1.2.2.1 SeMask

class SeMaskCls

SeMask commands group definition. 1 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1.2.2.2 Rbw

SCPI Command :

```
SENSe:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:RBW:USED
```

class RbwCls

Rbw commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class UsedStruct

Structure for reading output parameters. Fields:

- Trace_1: int: RBW for trace 1 (smallest RBW)
- Trace_2: int: RBW for trace 2 (intermediate RBW)
- Trace_3: int: RBW for trace 3 (largest RBW)

get_used() → UsedStruct

```
# SCPI: SENSe:LTE:MEASurement<Instance>:MEValuation:SPECtrum:SEMask:RBW:USED
value: UsedStruct = driver.sense.lteMeas.multiEval.spectrum.seMask.rbw.get_
↪used()
```

Queries the resolution bandwidths (RBW) allowed for spectrum emission measurements. The RBWs depend on the channel bandwidth and on the 'network signaled value'.

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

6.4 Trigger

class TriggerCls

Trigger commands group definition. 17 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.4.1 LteMeas

class LteMeasCls

LteMeas commands group definition. 17 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.4.1.1 MultiEval

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:MEValuation:THReshold
TRIGger:LTE:MEASurement<Instance>:MEValuation:SLOPe
TRIGger:LTE:MEASurement<Instance>:MEValuation:DELay
TRIGger:LTE:MEASurement<Instance>:MEValuation:TOUT
TRIGger:LTE:MEASurement<Instance>:MEValuation:MGAP
TRIGger:LTE:MEASurement<Instance>:MEValuation:SMODe
TRIGger:LTE:MEASurement<Instance>:MEValuation:AMODe
```

class MultiEvalCls

MultiEval commands group definition. 9 total commands, 1 Subgroups, 7 group commands

get_amode() → MevAcquisitionMode

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:AMODe
value: enums.MevAcquisitionMode = driver.trigger.lteMeas.multiEval.get_amode()
```

Selects whether the measurement synchronizes to a slot boundary or to a subframe boundary. The parameter is relevant for Free Run (Fast Sync) and for list mode measurements with Synchronization Mode = Enhanced.

return
acquisition_mode: No help available

get_delay() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:DELay
value: float = driver.trigger.lteMeas.multiEval.get_delay()
```

Defines a time delaying the start of the measurement relative to the trigger event. This setting has no influence on free run measurements.

return
delay: No help available

get_mgap() → int

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:MGAP
value: int = driver.trigger.lteMeas.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_trig_gap: No help available

get_slope() → SignalSlope

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SLOPe
value: enums.SignalSlope = driver.trigger.lteMeas.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: Rising edge FEDGe: Falling edge

get_smode() → SyncMode

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SMODe
value: enums.SyncMode = driver.trigger.lteMeas.multiEval.get_smode()
```

Selects the size of the search window for synchronization - normal or enhanced.

return
sync_mode: No help available

get_threshold() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:THReshold
value: float or bool = driver.trigger.lteMeas.multiEval.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return
trig_threshold: (float or boolean) No help available

get_timeout() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:TOUT
value: float or bool = driver.trigger.lteMeas.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: (float or boolean) No help available

set_amode(*acquisition_mode: MevAcquisitionMode*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:AMODe
driver.trigger.lteMeas.multiEval.set_amode(acquisition_mode = enums.
↪MevAcquisitionMode.SLOT)
```

Selects whether the measurement synchronizes to a slot boundary or to a subframe boundary. The parameter is relevant for Free Run (Fast Sync) and for list mode measurements with Synchronization Mode = Enhanced.

param acquisition_mode
No help available

set_delay(*delay: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:DELay
driver.trigger.lteMeas.multiEval.set_delay(delay = 1.0)
```

Defines a time delaying the start of the measurement relative to the trigger event. This setting has no influence on free run measurements.

param delay
No help available

set_mgap(*min_trig_gap: int*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:MGAP
driver.trigger.lteMeas.multiEval.set_mgap(min_trig_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_trig_gap
No help available

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SLOPe
driver.trigger.lteMeas.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: Rising edge FEDGe: Falling edge

set_smode(*sync_mode: SyncMode*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SMODe
driver.trigger.lteMeas.multiEval.set_smode(sync_mode = enums.SyncMode.ENHanced)
```

Selects the size of the search window for synchronization - normal or enhanced.

param sync_mode
No help available

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:THReshold
driver.trigger.lteMeas.multiEval.set_threshold(trig_threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param trig_threshold
(float or boolean) No help available

set_timeout(*trigger_timeout: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:TOUT
driver.trigger.lteMeas.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
(float or boolean) No help available

Cloning the Group

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

Subgroups

6.4.1.1.1 ListPy

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:MODE
TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:NBANdwidth
```

class ListPyCls

ListPy commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_mode() → ListMode

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:MODE
value: enums.ListMode = driver.trigger.lteMeas.multiEval.listPy.get_mode()
```

Specifies the trigger mode for list mode measurements. For configuration of retrigger flags, see method RsCMPX_LteMeas. Configure.LteMeas.MultiEval.ListPy.Segment.Setup.set. For configuration of the global trigger source, see TRIGger:LTE:MEAS<i>:MEValuation:SOURce.

return
mode: - 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 global trigger source is used. - SEGMent: The retrigger flag of each segment is evaluated. It defines whether a trigger event is required and which trigger source is used.

get_nbandwidth() → BandwidthNarrow

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:NBANdwidth
value: enums.BandwidthNarrow = driver.trigger.lteMeas.multiEval.listPy.get_
↪nbandwidth()
```

Selects the trigger evaluation bandwidth for the retrigger source IFPNarrowband. Select the retrigger source via method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Segment.Setup.set.

return
bandwidth: Evaluation bandwidth 10 MHz to 80 MHz

set_mode(*mode: ListMode*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:MODE
driver.trigger.lteMeas.multiEval.listPy.set_mode(mode = enums.ListMode.ONCE)
```

Specifies the trigger mode for list mode measurements. For configuration of retrigger flags, see method RsCMPX_LteMeas. Configure.LteMeas.MultiEval.ListPy.Segment.Setup.set. For configuration of the global trigger source, see TRIGger:LTE:MEAS<i>:MEValuation:SOURce.

param mode

- 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 global trigger source is used.
- SEGMent: The retrigger flag of each segment is evaluated. It defines whether a trigger event is required and which trigger source is used.

set_nbandwidth(*bandwidth: BandwidthNarrow*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:LIST:NBANdwidth
driver.trigger.lteMeas.multiEval.listPy.set_nbandwidth(bandwidth = enums.
↪BandwidthNarrow.M010)
```

Selects the trigger evaluation bandwidth for the retrigger source IFPNarrowband. Select the retrigger source via method RsCMPX_LteMeas.Configure.LteMeas.MultiEval.ListPy.Segment.Setup.set.

param bandwidth
Evaluation bandwidth 10 MHz to 80 MHz

6.4.1.2 Prach

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:PRACh:THReshold
TRIGger:LTE:MEASurement<Instance>:PRACh:SLOPe
TRIGger:LTE:MEASurement<Instance>:PRACh:TOUT
TRIGger:LTE:MEASurement<Instance>:PRACh:MGAP
```

class PrachCls

Prach commands group definition. 4 total commands, 0 Subgroups, 4 group commands

get_mgap() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:MGAP
value: float = driver.trigger.lteMeas.prach.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_trig_gap: No help available

get_slope() → SignalSlope

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SLOPe
value: enums.SignalSlope = driver.trigger.lteMeas.prach.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: Rising edge FEDGe: Falling edge

get_threshold() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:THReshold
value: float or bool = driver.trigger.lteMeas.prach.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return
trig_threshold: (float or boolean) No help available

get_timeout() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:TOUT
value: float or bool = driver.trigger.lteMeas.prach.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: (float or boolean) No help available

set_mgap(*min_trig_gap: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:MGAP
driver.trigger.lteMeas.prach.set_mgap(min_trig_gap = 1.0)
```

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_trig_gap
No help available

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SLOPe
driver.trigger.lteMeas.prach.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: Rising edge FEDGe: Falling edge

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:THReshold
driver.trigger.lteMeas.prach.set_threshold(trig_threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param trig_threshold
(float or boolean) No help available

set_timeout(*trigger_timeout: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:TOUT
driver.trigger.lteMeas.prach.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
(float or boolean) No help available

6.4.1.3 Srs

SCPI Commands :

```
TRIGger:LTE:MEASurement<Instance>:SRS:THReshold
TRIGger:LTE:MEASurement<Instance>:SRS:SLOPe
TRIGger:LTE:MEASurement<Instance>:SRS:TOUT
TRIGger:LTE:MEASurement<Instance>:SRS:MGAP
```

class SrsCls

Srs commands group definition. 4 total commands, 0 Subgroups, 4 group commands

get_mgap() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:MGAP
value: float = driver.trigger.lteMeas.srs.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_trig_gap: No help available

get_slope() → SignalSlope

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SLOPe
value: enums.SignalSlope = driver.trigger.lteMeas.srs.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: Rising edge FEDGe: Falling edge

get_threshold() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:THReshold
value: float or bool = driver.trigger.lteMeas.srs.get_threshold()
```

Defines the trigger threshold for power trigger sources.

return
trig_threshold: (float or boolean) No help available

get_timeout() → float

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:TOUT
value: float or bool = driver.trigger.lteMeas.srs.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: (float or boolean) No help available

set_mgap(*min_trig_gap: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:MGAP
driver.trigger.lteMeas.srs.set_mgap(min_trig_gap = 1.0)
```

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_trig_gap
No help available

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SLOPe
driver.trigger.lteMeas.srs.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: Rising edge FEDGe: Falling edge

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:THReshold
driver.trigger.lteMeas.srs.set_threshold(trig_threshold = 1.0)
```

Defines the trigger threshold for power trigger sources.

param trig_threshold
(float or boolean) No help available

set_timeout(*trigger_timeout: float*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:TOUT
driver.trigger.lteMeas.srs.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
(float or boolean) No help available

CHAPTER

SEVEN

RSCMPX_LTEMEAS UTILITIES

class Utilities

Common utility class. Utility functions common for all types of drivers.

Access snippet: `utils = RsCMPX_LteMeas.utilities`

property logger: *ScpiLogger*

Scpi Logger interface, see *here*

Access snippet: `logger = RsCMPX_LteMeas.utilities.logger`

property driver_version: str

Returns the instrument driver version.

property idn_string: str

Returns instrument's identification string - the response on the SCPI command *IDN?

property manufacturer: str

Returns manufacturer of the instrument.

property full_instrument_model_name: str

Returns the current instrument's full name e.g. 'FSW26'.

property instrument_model_name: str

Returns the current instrument's family name e.g. 'FSW'.

property supported_models: List[str]

Returns a list of the instrument models supported by this instrument driver.

property instrument_firmware_version: str

Returns instrument's firmware version.

property instrument_serial_number: str

Returns instrument's serial_number.

query_opc(*timeout: int = 0*) → int

SCPI command: *OPC? Queries the instrument's OPC bit and hence it waits until the instrument reports operation complete. If you define timeout > 0, the VISA timeout is set to that value just for this method call.

property instrument_status_checking: bool

Sets / returns Instrument Status Checking. When True (default is True), all the driver methods and properties are sending "SYSTem:ERRor?" at the end to immediately react on error that might have occurred. We recommend to keep the state checking ON all the time. Switch it OFF only in rare cases when you require maximum speed. The default state after initializing the session is ON.

property encoding: str

Returns string<=>bytes encoding of the session.

property opc_query_after_write: bool

Sets / returns Instrument *OPC? query sending after each command write. When True, (default is False) the driver sends *OPC? every time a write command is performed. Use this if you want to make sure your sequence is performed command-after-command.

property bin_float_numbers_format: BinFloatFormat

Sets / returns format of float numbers when transferred as binary data.

property bin_int_numbers_format: BinIntFormat

Sets / returns format of integer numbers when transferred as binary data.

clear_status() → None

Clears instrument's status system, the session's I/O buffers and the instrument's error queue.

query_all_errors() → List[str]

Queries and clears all the errors from the instrument's error queue. The method returns list of strings as error messages. If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty. If you want to include the error codes, call the query_all_errors_with_codes()

query_all_errors_with_codes() → List[Tuple[int, str]]

Queries and clears all the errors from the instrument's error queue. The method returns list of tuples (code: int, message: str). If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty.

property instrument_options: List[str]

Returns all the instrument options. The options are sorted in the ascending order starting with K-options and continuing with B-options.

reset() → None

SCPI command: *RST Sends *RST command + calls the clear_status().

default_instrument_setup() → None

Custom steps performed at the init and at the reset().

self_test(*timeout: int = None*) → Tuple[int, str]

SCPI command: *TST? Performs instrument's self-test. Returns tuple (code:int, message: str). Code 0 means the self-test passed. You can define the custom timeout in milliseconds. If you do not define it, the default selftest timeout is used (usually 60 secs).

is_connection_active() → bool

Returns true, if the VISA connection is active and the communication with the instrument still works.

reconnect(*force_close: bool = False*) → bool

If the connection is not active, the method tries to reconnect to the device If the connection is active, and force_close is False, the method does nothing. If the connection is active, and force_close is True, the method closes, and opens the session again. Returns True, if the reconnection has been performed.

property resource_name: int

Returns the resource name used in the constructor

property opc_timeout: int

Sets / returns timeout in milliseconds for all the operations that use OPC synchronization.

property visa_timeout: int

Sets / returns visa IO timeout in milliseconds.

property data_chunk_size: int

Sets / returns the maximum size of one block transferred during write/read operations

property visa_manufacturer: int

Returns the manufacturer of the current VISA session.

process_all_commands() → None

SCPI command: *WAI Stops further commands processing until all commands sent before *WAI have been executed.

write_str(*cmd: str*) → None

Writes the command to the instrument.

write(*cmd: str*) → None

This method is an alias to the write_str(). Writes the command to the instrument as string.

write_int(*cmd: str*, *param: int*) → None

Writes the command to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2'

write_int_with_opc(*cmd: str*, *param: int*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2' If you do not provide timeout, the method uses current opc_timeout.

write_float(*cmd: str*, *param: float*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6'

write_float_with_opc(*cmd: str*, *param: float*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6' If you do not provide timeout, the method uses current opc_timeout.

write_bool(*cmd: str*, *param: bool*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON'

write_bool_with_opc(*cmd: str*, *param: bool*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON' If you do not provide timeout, the method uses current opc_timeout.

query_str(*query: str*) → str

Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query(*query: str*) → str

This method is an alias to the query_str(). Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query_bool(*query: str*) → bool

Sends the query to the instrument and returns the response as boolean.

query_int(*query: str*) → int

Sends the query to the instrument and returns the response as integer.

query_float(*query: str*) → float

Sends the query to the instrument and returns the response as float.

write_str_with_opc(*cmd: str*, *timeout: int = None*) → None

Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

write_with_opc(*cmd: str*, *timeout: int = None*) → None

This method is an alias to the write_str_with_opc(). Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

query_str_with_opc(*query: str*, *timeout: int = None*) → str

Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_with_opc(*query: str*, *timeout: int = None*) → str

This method is an alias to the query_str_with_opc(). Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_bool_with_opc(*query: str*, *timeout: int = None*) → bool

Sends the opc-synced query to the instrument and returns the response as boolean. If you do not provide timeout, the method uses current opc_timeout.

query_int_with_opc(*query: str*, *timeout: int = None*) → int

Sends the opc-synced query to the instrument and returns the response as integer. If you do not provide timeout, the method uses current opc_timeout.

query_float_with_opc(*query: str*, *timeout: int = None*) → float

Sends the opc-synced query to the instrument and returns the response as float. If you do not provide timeout, the method uses current opc_timeout.

write_bin_block(*cmd: str*, *payload: bytes*) → None

Writes all the payload as binary data block to the instrument. The binary data header is added at the beginning of the transmission automatically, do not include it in the payload!!!

query_bin_block(*query: str*) → bytes

Queries binary data block to bytes. Throws an exception if the returned data was not a binary data. Returns data:bytes

query_bin_block_with_opc(*query: str*, *timeout: int = None*) → bytes

Sends a OPC-synced query and returns binary data block to bytes. If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_float_list(*query: str*) → List[float]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_float_list_with_opc(*query: str*, *timeout: int = None*) → List[float]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_int_list(*query: str*) → List[int]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_int_list_with_opc(*query: str*, *timeout: int = None*) → List[int]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_block_to_file(*query: str*, *file_path: str*, *append: bool = False*) → None

Queries binary data block to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data. Example for transferring a file from Instrument -> PC: query = f'MMEM:DATA? '{INSTR_FILE_PATH}'". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

query_bin_block_to_file_with_opc(*query: str*, *file_path: str*, *append: bool = False*, *timeout: int = None*) → None

Sends a OPC-synced query and writes the returned data to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data.

write_bin_block_from_file(*cmd: str*, *file_path: str*) → None

Writes data from the file as binary data block to the instrument using the provided command. Example for transferring a file from PC -> Instrument: cmd = f'MMEM:DATA '{INSTR_FILE_PATH}',". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

send_file_from_pc_to_instrument(*source_pc_file: str*, *target_instr_file: str*) → None

SCPI Command: MMEM:DATA

Sends file from PC to the instrument

read_file_from_instrument_to_pc(*source_instr_file: str*, *target_pc_file: str*, *append_to_pc_file: bool = False*) → None

SCPI Command: MMEM:DATA?

Reads file from instrument to the PC.

Set the `append_to_pc_file` to True if you want to append the read content to the end of the existing PC file

get_last_sent_cmd() → str

Returns the last commands sent to the instrument. Only works in simulation mode

go_to_local() → None

Puts the instrument into local state.

go_to_remote() → None

Puts the instrument into remote state.

get_lock() → RLock

Returns the thread lock for the current session.

By default:

- If you create standard new RsCMPX_LteMeas instance with new VISA session, the session gets a new thread lock. You can assign it to other RsCMPX_LteMeas sessions in order to share one physical instrument with a multi-thread access.
- If you create new RsCMPX_LteMeas from an existing session, the thread lock is shared automatically making both instances multi-thread safe.

You can always assign new thread lock by calling `driver.utilities.assign_lock()`

assign_lock(*lock: RLock*) → None

Assigns the provided thread lock.

clear_lock()

Clears the existing thread lock, making the current session thread-independent from others that might share the current thread lock.

sync_from(*source: Utilities*) → None

Synchronises these Utils with the source.

CHAPTER

EIGHT

RSCMPX_LTEMEAS LOGGER

Check the usage in the Getting Started chapter *here*.

class ScpiLogger

Base class for SCPI logging

mode

Sets the logging ON or OFF. Additionally, you can set the logging ON only for errors. Possible values:

- LoggingMode.Off - logging is switched OFF
- LoggingMode.On - logging is switched ON
- LoggingMode.Errors - logging is switched ON, but only for error entries
- LoggingMode.Default - sets the logging to default - the value you have set with logger.default_mode

default_mode

Sets / returns the default logging mode. You can recall the default mode by calling the logger.mode = LoggingMode.Default.

Data Type
LoggingMode

device_name: str

Use this property to change the resource name in the log from the default Resource Name (e.g. TCPIP::192.168.2.101::INSTR) to another name e.g. 'MySigGen1'.

set_logging_target(*target*, *console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target - the target must implement write() and flush(). You can optionally set the console and UDP logging ON or OFF. This method switches the logging target global OFF.

get_logging_target()

Based on the global_mode, it returns the logging target: either the local or the global one.

set_logging_target_global(*console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target to global. The global target must be defined. You can optionally set the console and UDP logging ON or OFF.

log_to_console

Returns logging to console status.

log_to_udp

Returns logging to UDP status.

log_to_console_and_udp

Returns true, if both logging to UDP and console in are True.

info_raw(*log_entry: str*, *add_new_line: bool = True*) → None

Method for logging the raw string without any formatting.

info(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one info entry. For binary log_string, use the info_bin()

error(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one error entry.

set_relative_timestamp(*timestamp: datetime*) → None

If set, the further timestamps will be relative to the entered time.

set_relative_timestamp_now() → None

Sets the relative timestamp to the current time.

get_relative_timestamp() → datetime

Based on the global_mode, it returns the relative timestamp: either the local or the global one.

clear_relative_timestamp() → None

Clears the reference time, and the further logging continues with absolute times.

flush() → None

Flush all the entries.

log_status_check_ok

Sets / returns the current status of status checking OK. If True (default), the log contains logging of the status checking 'Status check: OK'. If False, the 'Status check: OK' is skipped - the log is more compact. Errors will still be logged.

clear_cached_entries() → None

Clears potential cached log entries. Cached log entries are generated when the Logging is ON, but no target has been defined yet.

set_format_string(*value: str*, *line_divider: str = '\n'*) → None

Sets new format string and line divider. If you just want to set the line divider, set the format string value=None The original format string is: `PAD_LEFT12(%START_TIME%) PAD_LEFT25(%DEVICE_NAME%) PAD_LEFT12(%DURATION%) %LOG_STRING_INFO%: %LOG_STRING%`

restore_format_string() → None

Restores the original format string and the line divider to LF

abbreviated_max_len_ascii: int

Defines the maximum length of one ASCII log entry. Default value is 200 characters.

abbreviated_max_len_bin: int

Defines the maximum length of one Binary log entry. Default value is 2048 bytes.

abbreviated_max_len_list: int

Defines the maximum length of one list entry. Default value is 100 elements.

bin_line_block_size: int

Defines number of bytes to display in one line. Default value is 16 bytes.

udp_port

Returns udp logging port.

target_auto_flushing

Returns status of the auto-flushing for the logging target.

CHAPTER

NINE

RSCMPX_LTEMEAS EVENTS

Check the usage in the Getting Started chapter *here*.

class Events

Common Events class. Event-related methods and properties. Here you can set all the event handlers.

property before_query_handler: Callable

Returns the handler of before_query events.

Returns
current `before_query_handler`

property before_write_handler: Callable

Returns the handler of before_write events.

Returns
current `before_write_handler`

property io_events_include_data: bool

Returns the current state of the io_events_include_data See the setter for more details.

property on_read_handler: Callable

Returns the handler of on_read events.

Returns
current `on_read_handler`

property on_write_handler: Callable

Returns the handler of on_write events.

Returns
current `on_write_handler`

sync_from(*source: Events*) → None

Synchronises these Events with the source.

CHAPTER

TEN

INDEX

INDEX

CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage, 342
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent, 343
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EXTReme, 344
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:AVERage, 346
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:CURRent, 347
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:EXTReme, 348
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:SDEViation, 349
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:STDDev, 350
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 351
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:STDDev, 353
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:AVERage, 354
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:CURRent, 354
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:EXTReme, 356
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:SDEViation, 357
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:STDDev, 357
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 359
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:STDDev, 360
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage, 360
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent, 362
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EXTReme, 363
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:AVERage, 363
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:CURRent, 365
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:EXTReme, 366
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:SDEViation, 367
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:STDDev, 369
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 369
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:STDDev, 370
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage, 372
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent, 373
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EXTReme, 374
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:AVERage, 375
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:CURRent, 376
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:EXTReme, 377
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:SDEViation, 379
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:STDDev, 380
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 380
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:STDDev, 382
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:AVERage, 383
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:CURRent, 383
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:EXTReme, 385
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:SDEViation, 386
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation[:RMS]:STDDev, 387
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 389
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:STDDev, 389
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:AVERage, 390
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:CURRent, 392
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:EXTReme, 393
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:AVERage, 394
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:CURRent, 395
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:EXTReme, 396
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:SDEViation, 397
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:STDDev, 399
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:SDEViation, 400
CALCulate:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:STDDev, 400

CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:MODulation
CONFigure:LTE:MEASurement<Instance>:MEValuation:MOExceptio
CONFigure:LTE:MEASurement<Instance>:MEValuation:MSLot,
CONFigure:LTE:MEASurement<Instance>:MEValuation:MSUBframes
CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue,
CONFigure:LTE:MEASurement<Instance>:MEValuation:NSValue:CA
CONFigure:LTE:MEASurement<Instance>:MEValuation:NVFilter,
CONFigure:LTE:MEASurement<Instance>:MEValuation:ORVFilter,
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:
CONFigure:LTE:MEASurement<Instance>:MEValuation:PDYNamics:
CONFigure:LTE:MEASurement<Instance>:MEValuation:PFORmat,
CONFigure:LTE:MEASurement<Instance>:MEValuation:POWer:HDMo
CONFigure:LTE:MEASurement<Instance>:MEValuation:PSEarch,
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocati
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocati
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocati
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocati
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocati
CONFigure:LTE:MEASurement<Instance>:MEValuation:RBALlocati

D

E

F

FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:HIGH:CURRent, 393
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:HIGH:EXTReme, 394
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:HIGH:SDEViation, 394
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:LOW:AVERage, 395
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:LOW:CURRent, 396
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:LOW:EXTReme, 397
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:PEAK:LOW:SDEViation, 398
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:MODulation:RMS:HIGH:AVERage, 399
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:RMS:HIGH:CURRent, 400
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:PMONitor:RMS:HIGH:EXTReme, 400
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXP:RMS:HIGH:SDEViation, 401
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXP:RMS:LOW:AVERage, 402
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXP:RMS:LOW:CURRent, 403
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXP:RMS:LOW:EXTReme, 403
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:POWer:TXP:RMS:LOW:SDEViation, 404
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:POWer:AVERage, 405
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:POWer:CURRent, 406
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:POWer:MAXimum, 406
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:POWer:MINimum, 407
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:POWer:SDEViation, 408
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:PSD:AVERage, 409
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:PSD:CURRent, 410
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:PSD:MAXimum, 410
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:PSD:MINimum, 411
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:PSD:SDEViation, 412
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:SCHType, 412
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:TIME:AVERage, 413
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:TIME:CURRent, 414
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:TIME:EXTReme, 415
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:TIME:SDEViation, 416
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent<Nr>:TYPE, 416
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TIME:AVERage, 417
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TIME:CURRent, 418
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TIME:EXTReme, 419
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TIME:SDEViation, 420
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TRACe:AVERage, 420
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TRACe:CURRent, 421
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TRACe:EXTReme, 422
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TRACe:SDEViation, 423
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:TYPE, 424
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:AVERage, 424
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:CURRent, 425
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:EXTReme, 426
FETCh:LTE:MEASurement<Instance>:MEValuation:MODulation:SDEViation, 428
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:RMS:AVERage, 429
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:RMS:CURRent, 430
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:RMS:EXTReme, 431
FETCh:LTE:MEASurement<Instance>:MEValuation:PMONitor:RMS:SDEViation, 432
FETCh:LTE:MEASurement<Instance>:MEValuation:POWer:AVERage, 433
FETCh:LTE:MEASurement<Instance>:MEValuation:POWer:CURRent, 435
FETCh:LTE:MEASurement<Instance>:MEValuation:POWer:MAXimum, 436
FETCh:LTE:MEASurement<Instance>:MEValuation:POWer:MINimum, 437
FETCh:LTE:MEASurement<Instance>:MEValuation:POWer:SDEViation, 440
FETCh:LTE:MEASurement<Instance>:MEValuation:TRACe:AVERage, 441

FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:MARGin:CURRent:POSition:MERRor:MAXimum
499
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:MARGin:MINimum:AVERage:MODulation:AVE
500
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:MARGin:MINimum:POSition:MODulation:CUR
501
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SDEViation:MEValuation:MODulation:DAL
502
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:MODulation:DCH
503
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:MODulation:DMO
504
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:MODulation:EXT
505
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:MODulation:SCH
506
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:MODulation:SDE
507
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PDYNamics:AVER
508
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PDYNamics:CURR
508
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PDYNamics:MAXi
509
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PDYNamics:MINi
510
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PDYNamics:SDEV
511
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PERRor:AVERage
511
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PERRor:AVERage
512
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PERRor:CURRent
513
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PERRor:CURRent
514
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PERRor:MAXimum
514
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PERRor:MAXimum
515
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PMONitor:AVERa
516
FETCh:LTE:MEASurement<Instance>:MEValuation:LIST:SEGMent:MEASurement:SEMask:MARGin:AREA<Area>:MEValuation:PMONitor:CC<Nr
516
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr
517
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:AVERage:MEASUREF:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr
519
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr
520
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:CURRent:MEASUREF:MEASurement<Instance>:MEValuation:PMONitor:CC<Nr
521
FETCh:LTE:MEASurement<Instance>:MEValuation:MERRor:MAXimum:MEASurement<Instance>:MEValuation:PMONitor:CURRe
522

G

I

L

M

R