RsCMPX_NrFr1Meas

Release 5.0.80.17

Rohde & Schwarz

Apr 19, 2024

CONTENTS:


5G
CMX/P

CHAPTER

ONE

REVISION HISTORY

1.1 RsCMPX_NrFr1Meas

Rohde & Schwarz CMX/CMP New Radio FR1 Measurement RsCMPX_NrFr1Meas instrument driver.

Basic Hello-World code:

```
from RsCMPX_NrFr1Meas import *

instr = RsCMPX_NrFr1Meas('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-NrFr1Meas/

Documentation: https://RsCMPX-NrFr1Meas.readthedocs.io/

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

1.1.1 Version history

Latest release notes summary: Update for FW 5.0.80

Version 5.0.80

- Update for FW 5.0.80

Version 4.0.186

- Fixed documentation

Version 4.0.185

- Update to FW 4.0.185

Version 4.0.140

- Update of RsCMPX_NrFr1Meas to FW 4.0.140 from the complete FW package 7.10.0

Version 4.0.60

- Update of RsCMPX_NrFr1Meas to FW 4.0.60

Version 4.0.10

- First released version

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

RsCMPX_NrFr1Meas 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_NrFr1Meas 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_NrFr1Meas`

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_NrFr1Meas` 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_NrFr1Meas offline:

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCMPX_NrFr1Meas needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCMPX_NrFr1Meas package to your computer from the pypi.org: https://pypi.org/project/RsCMPX_NrFr1Meas/#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_NrFr1Meas-5.0.80.17.tar`

2.3 Finding Available Instruments

Like the pyvisa's ResourceManager, the RsCMPX_NrFr1Meas can search for available instruments:

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

from RsCMPX_NrFr1Meas import *

# Use the instr_list string items as resource names in the RsCMPX_NrFr1Meas constructor
instr_list = RsCMPX_NrFr1Meas.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_NrFr1Meas 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_NrFr1Meas.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_NrFr1Meas 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_NrFr1Meas 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_NrFr1Meas module for remote-controlling your␣
↪instrument
Preconditions:

- Installed RsCMPX_NrFr1Meas Python module Version 5.0.80 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCMPX_NrFr1Meas import *

# A good practice is to assure that you have a certain minimum version installed
RsCMPX_NrFr1Meas.assert_minimum_version('5.0.80')
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_NrFr1Meas(resource_string_1)

idn = driver.utilities.query_str('*IDN?')
print(f"\nHello, I am: '{idn}'")
print(f'RsCMPX_NrFr1Meas 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_NrFr1Meas 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_NrFr1Meas('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_NrFr1Meas 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_NrFr1Meas allows you to choose which VISA to use:

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

from RsCMPX_NrFr1Meas import *

# Force use of the Rs Visa. For NI Visa, use the "SelectVisa='ni'"
driver = RsCMPX_NrFr1Meas('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_NrFr1Meas has something for you too - **no Visa installation raw LAN socket**:

```
"""
Using RsCMPX_NrFr1Meas without VISA for LAN Raw socket communication
"""
```

(continues on next page)

(continued from previous page)

```
from RsCMPX_NrFr1Meas import *

driver = RsCMPX_NrFr1Meas('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_NrFr1Meas('TCPIP::192.168.56.101::hislip0', True, True, "Simulate=True")
```

More `option_string` tokens are separated by comma:

```
driver = RsCMPX_NrFr1Meas('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_NrFr1Meas objects:

```
"""
Sharing the same physical VISA session by two different RsCMPX_NrFr1Meas objects
"""

from RsCMPX_NrFr1Meas import *

driver1 = RsCMPX_NrFr1Meas('TCPIP::192.168.56.101::INSTR', True, True)
driver2 = RsCMPX_NrFr1Meas.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_NrFr1Meas 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_NrFr1Meas import *

driver = RsCMPX_NrFr1Meas('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_NrFr1Meas import *

driver = RsCMPX_NrFr1Meas('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_NrFr1Meas raises an exception. Speaking of exceptions, an important feature of the RsCMPX_NrFr1Meas 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_NrFr1Meas import *

driver = RsCMPX_NrFr1Meas('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_NrFr1Meas 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_NrFr1Meas 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_NrFr1Meas import *

driver = None
# Try-catch for initialization. If an error occures, the ResourceError is raised
try:
    driver = RsCMPX_NrFr1Meas('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_NrFr1Meas exceptions
    print(e.args[0])
    print('Some other RsCMPX_NrFr1Meas 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_NrFr1Meas, 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_NrFr1Meas 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_NrFr1Meas has this covered. And, this feature is quite universal - not just for big files transfer, but for any data in both directions.

RsCMPX_NrFr1Meas 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_NrFr1Meas 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_NrFr1Meas('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_NrFr1Meas 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_NrFr1Meas 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_NrFr1Meas object
"""

import threading
from RsCMPX_NrFr1Meas import *
```

(continues on next page)

(continued from previous page)

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


driver = RsCMPX_NrFr1Meas('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_NrFr1Meas objects with shared session
"""

import threading
from RsCMPX_NrFr1Meas import *


def execute(session: RsCMPX_NrFr1Meas, 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_NrFr1Meas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_NrFr1Meas.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()
```

(continues on next page)

(continued from previous page)

```
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()
```

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_NrFr1Meas takes care of it for you. The text below describes this scenario.

Run the following example:

```
"""
Multiple threads are accessing two RsCMPX_NrFr1Meas objects with two separate sessions
"""

import threading
from RsCMPX_NrFr1Meas import *


def execute(session: RsCMPX_NrFr1Meas, 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_NrFr1Meas('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_NrFr1Meas('TCPIP::192.168.56.101::INSTR')
```

(continues on next page)

(continued from previous page)

```
driver1.utilities.visa_timeout = 200
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_NrFr1Meas import *
```

(continues on next page)

(continued from previous page)

```
driver = RsCMPX_NrFr1Meas('TCPIP::192.168.1.101::INSTR')

# 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_NrFr1Meas('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_NrFr1Meas import *

driver = RsCMPX_NrFr1Meas('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_NrFr1Meas import *

driver = RsCMPX_NrFr1Meas('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 AllocatedSlots

```
# Example value:
value = enums.AllocatedSlots.ALL
# All values (1x):
ALL
```

3.2 Band

```
# First value:
value = enums.Band.OB1
# Last value:
value = enums.Band.OB99
# All values (60x):
OB1 | OB100 | OB101 | OB104 | OB12 | OB13 | OB14 | OB18
OB2 | OB20 | OB24 | OB25 | OB255 | OB256 | OB26 | OB28
OB3 | OB30 | OB34 | OB38 | OB39 | OB40 | OB41 | OB46
OB47 | OB48 | OB5 | OB50 | OB51 | OB53 | OB65 | OB66
OB7 | OB70 | OB71 | OB74 | OB75 | OB76 | OB77 | OB78
OB79 | OB8 | OB80 | OB81 | OB82 | OB83 | OB84 | OB85
OB86 | OB89 | OB90 | OB91 | OB92 | OB93 | OB94 | OB95
OB96 | OB97 | OB98 | OB99
```

3.3 BandwidthPart

```
# Example value:
value = enums.BandwidthPart.BWP0
# All values (1x):
BWP0
```

3.4 CarrierComponent

```
# Example value:
value = enums.CarrierComponent.CC1
# All values (2x):
CC1 | CC2
```

3.5 CarrierPosition

```
# Example value:
value = enums.CarrierPosition.LONR
# All values (2x):
LONR | RONR
```

3.6 ChannelBwidth

```
# First value:
value = enums.ChannelBwidth.B005
# Last value:
value = enums.ChannelBwidth.B100
# All values (15x):
B005 | B010 | B015 | B020 | B025 | B030 | B035 | B040
B045 | B050 | B060 | B070 | B080 | B090 | B100
```

3.7 ChannelBwidthB

```
# Example value:
value = enums.ChannelBwidthB.B005
# All values (4x):
B005 | B010 | B015 | B020
```

3.8 ChannelTypeA

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

3.9 ChannelTypeB

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

3.10 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
```

3.11 ConfigType

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

3.12 CyclicPrefix

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

3.13 DmrsInit

```
# Example value:
value = enums.DmrsInit.CID
# All values (2x):
CID | DID
```

3.14 DmrsPort

```
# Example value:
value = enums.DmrsPort.ALL
# All values (3x):
ALL | P1000 | P1001
```

3.15 DuplexModeB

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

3.16 Generator

```
# Example value:
value = enums.Generator.DID
# All values (2x):
DID | PHY
```

3.17 GhopingInit

```
# Example value:
value = enums.GhopingInit.CID
# All values (2x):
CID | HID
```

3.18 GroupHopping

```
# Example value:
value = enums.GroupHopping.DISable
# All values (3x):
DISable | ENABle | NEITher
```

3.19 Ktc

```
# Example value:
value = enums.Ktc.N2
# All values (3x):
N2 | N4 | N8
```

3.20 Lagging

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

3.21 Leading

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

3.22 ListMode

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

3.23 LowHigh

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

3.24 MappingType

```
# Example value:
value = enums.MappingType.A
# All values (2x):
A | B
```

3.25 MaxLength

```
# Example value:
value = enums.MaxLength.DOUBle
# All values (2x):
DOUBle | SINGle
```

3.26 MeasFilter

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

3.27 MeasurementMode

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

3.28 MeasureSlot

```
# Example value:
value = enums.MeasureSlot.ALL
# All values (6x):
ALL | MS0 | MS1 | MS2 | MS3 | UDEF
```

3.29 MevLimit

```
# Example value:
value = enums.MevLimit.STD
# All values (2x):
STD | UDEF
```

3.30 Modulation

```
# Example value:
value = enums.Modulation.BPSK
# All values (6x):
BPSK | BPWS | Q16 | Q256 | Q64 | QPSK
```

3.31 ModulationScheme

```
# Example value:
value = enums.ModulationScheme.AUTO
# All values (7x):
AUTO | BPSK | BPWS | Q16 | Q256 | Q64 | QPSK
```

3.32 ModulationSchemeB

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

3.33 NbTrigger

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

3.34 NetworkSigVal

```
# First value:
value = enums.NetworkSigVal.NS01
# Last value:
value = enums.NetworkSigVal.NSU43
# All values (103x):
NS01 | NS02 | NS03 | NS04 | NS05 | NS06 | NS07 | NS08
NS09 | NS10 | NS100 | NS11 | NS12 | NS13 | NS14 | NS15
NS16 | NS17 | NS18 | NS19 | NS20 | NS21 | NS22 | NS23
NS24 | NS25 | NS26 | NS27 | NS28 | 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 | NSU03 | NSU05 | NSU43
```

3.35 NumberSymbols

```
# Example value:
value = enums.NumberSymbols.N1
# All values (7x):
N1 | N10 | N12 | N14 | N2 | N4 | N8
```

3.36 ParameterSetMode

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

3.37 Periodicity

```
# First value:
value = enums.Periodicity.MS05
# Last value:
value = enums.Periodicity.MS5
# All values (9x):
MS05 | MS1 | MS10 | MS125 | MS2 | MS25 | MS3 | MS4
MS5
```

3.38 PeriodPreamble

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

3.39 PhaseComp

```
# Example value:
value = enums.PhaseComp.CAF
# All values (3x):
CAF | OFF | UDEF
```

3.40 PreambleFormat

```
# First value:
value = enums.PreambleFormat.PF0
# Last value:
value = enums.PreambleFormat.PFC2
# All values (13x):
PF0 | PF1 | PF2 | PF3 | PFA1 | PFA2 | PFA3 | PFB1
PFB2 | PFB3 | PFB4 | PFC0 | PFC2
```

3.41 PucchFormat

```
# Example value:
value = enums.PucchFormat.F0
# All values (5x):
F0 | F1 | F2 | F3 | F4
```

3.42 RbwA

```
# Example value:
value = enums.RbwA.K030
# All values (3x):
K030 | M1 | PC1
```

3.43 RbwB

```
# Example value:
value = enums.RbwB.K030
# All values (6x):
K030 | K100 | K400 | M1 | PC1 | PC2
```

3.44 RbwC

```
# Example value:
value = enums.RbwC.K030
# All values (3x):
K030 | K400 | M1
```

3.45 Repeat

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

3.46 ResourceState

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

3.47 RestrictedSet

```
# Example value:
value = enums.RestrictedSet.URES
# All values (1x):
URES
```

3.48 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
```

3.49 RetriggerFlag

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

3.50 Sharing

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

3.51 SignalPath

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

3.52 SignalSlope

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

3.53 SignalType

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

3.54 SrsPeriodicity

```
# First value:
value = enums.SrsPeriodicity.SL1
# Last value:
value = enums.SrsPeriodicity.SL80
# All values (17x):
SL1 | SL10 | SL1280 | SL16 | SL160 | SL2 | SL20 | SL2560
SL32 | SL320 | SL4 | SL40 | SL5 | SL64 | SL640 | SL8
SL80
```

3.55 StopCondition

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

3.56 SubCarrSpacing

```
# Example value:
value = enums.SubCarrSpacing.S15K
# All values (3x):
S15K | S30K | S60K
```

3.57 SubCarrSpacingB

```
# Example value:
value = enums.SubCarrSpacingB.S15K
# All values (5x):
S15K | S1K2 | S30K | S5K | S60K
```

3.58 SubChanSize

```
# Example value:
value = enums.SubChanSize.RB10
# All values (8x):
RB10 | RB100 | RB12 | RB15 | RB20 | RB25 | RB50 | RB75
```

3.59 SyncMode

```
# Example value:
value = enums.SyncMode.ENHanced
# All values (4x):
ENHanced | ESSLot | NORMal | NSSLot
```

3.60 TargetStateA

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

3.61 TargetSyncState

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

3.62 TimeMask

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

3.63 TraceSelect

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

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 AbsoluteMarker

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

4.3 AddTable

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

4.4 Allocation

```
# First value:
value = repcap.Allocation.Nr1
# Values (1x):
Nr1
```

4.5 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
```

4.6 AreaReduced

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

4.7 CarrierComponent

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

4.8 CarrierComponentFour

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

4.9 CarrierComponentOne

```
# First value:
value = repcap.CarrierComponentOne.Nr1
# Values (1x):
Nr1
```

4.10 ChannelBw

```
# First value:
value = repcap.ChannelBw.Bw5
# Range:
Bw5 .. Bw100
# All values (15x):
Bw5 | Bw10 | Bw15 | Bw20 | Bw25 | Bw30 | Bw35 | Bw40
Bw45 | Bw50 | Bw60 | Bw70 | Bw80 | Bw90 | Bw100
```

4.11 DeltaMarker

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

4.12 Difference

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

4.13 EonPower

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

4.14 EonPowerScs

```
# First value:
value = repcap.EonPowerScs.Nr15
# Values (3x):
Nr15 | Nr30 | Nr60
```

4.15 Layer

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

4.16 Maximum

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

4.17 MaxRange

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

4.18 Minimum

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

4.19 MinRange

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

4.20 PFormat

```
# First value:
value = repcap.PFormat.Nr1
# Range:
Nr1 .. Nr13
# All values (13x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13
```

4.21 Preamble

```
# First value:
value = repcap.Preamble.Nr1
# Range:
Nr1 .. Nr64
# All values (64x):
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
```

4.22 Qam

```
# First value:
value = repcap.Qam.Order16
# Values (3x):
Order16 | Order64 | Order256
```

4.23 Rbw

```
# First value:
value = repcap.Rbw.Bw1
# Range:
Bw1 .. Bw1000
# All values (6x):
Bw1 | Bw2 | Bw30 | Bw100 | Bw400 | Bw1000
```

4.24 Ripple

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

4.25 SEGMent

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

(continues on next page)

(continued from previous page)

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

4.26 UtraChannel

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

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_NRFR1MEAS API STRUCTURE

Global RepCaps

```
driver = RsCMPX_NrFr1Meas('TCPIP::192.168.2.101::hislip0')
# Instance range: Inst1 .. Inst16
rc = driver.repcap_instance_get()
driver.repcap_instance_set(repcap.Instance.Inst1)
```

class RsCMPX_NrFr1Meas(*resource_name: str*, *id_query: bool = True*, *reset: bool = False*, *options: str = None*, *direct_session: object = None*)

1055 total commands, 4 Subgroups, 0 group commands

Initializes new RsCMPX_NrFr1Meas 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_NrFr1Meas.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_NrFr1Meas session.

classmethod from_existing_session(*session: object*, *options: str = None*) → RsCMPX_NrFr1Meas

Creates a new RsCMPX_NrFr1Meas 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. 315 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 NrSubMeas

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:NANTenna
CONFigure:NRSub:MEASurement<Instance>:STYPe
CONFigure:NRSub:MEASurement<Instance>:BAND
CONFigure:NRSub:MEASurement<Instance>:SPATh
CONFigure:NRSub:MEASurement<Instance>:NCARrier
```

class NrSubMeasCls

NrSubMeas commands group definition. 315 total commands, 11 Subgroups, 5 group commands

get_band() → Band

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

Selects the operating band (OB) . The allowed input range depends on the duplex mode (FDD or TDD) .

return

band: TDD UL: OB34 | OB38 | … | OB41 | OB46 | OB47 | OB48 | OB50 | OB51 | OB53 | OB77 | … | OB84 | OB86 | OB89 | OB90 | OB95 | … | OB99 | OB101 | OB104

get_nantenna() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NANTenna
value: int = driver.configure.nrSubMeas.get_nantenna()
```

Selects the number of UE TX antennas to be measured.

return

number: No help available

get_ncarrier() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NCARrier
value: int = driver.configure.nrSubMeas.get_ncarrier()
```

Configures the number of contiguously aggregated UL carriers in the measured signal.

return
number: No help available

get_spath() → SignalPath

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SPATh
value: enums.SignalPath = driver.configure.nrSubMeas.get_spath()
```

No command help available

return
path: No help available

get_stype() → SignalType

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

No command help available

return
signal_type: No help available

set_band(*band: Band*) → None

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

Selects the operating band (OB) . The allowed input range depends on the duplex mode (FDD or TDD) .

param band
TDD UL: OB34 | OB38 | … | OB41 | OB46 | OB47 | OB48 | OB50 | OB51 | OB53 | OB77 | … | OB84 | OB86 | OB89 | OB90 | OB95 | … | OB99 | OB101 | OB104

set_nantenna(*number: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NANTenna
driver.configure.nrSubMeas.set_nantenna(number = 1)
```

Selects the number of UE TX antennas to be measured.

param number
No help available

set_ncarrier(*number: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NCARrier
driver.configure.nrSubMeas.set_ncarrier(number = 1)
```

Configures the number of contiguously aggregated UL carriers in the measured signal.

param number
No help available

set_spath(*path: SignalPath*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SPATh
driver.configure.nrSubMeas.set_spath(path = enums.SignalPath.NETWork)
```

No command help available

param path
No help available

`set_stype`(*signal_type: SignalType*) → None

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

No command help available

param signal_type
No help available

Cloning the Group

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

Subgroups

6.1.1.1 BwConfig

class BwConfigCls

BwConfig 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.nrSubMeas.bwConfig.clone()
```

Subgroups

6.1.1.1.1 Invoke

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:BWConfig:INVoke
```

class InvokeCls

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

class GetStruct

Response structure. Fields:

- Sub_Carr_Spacing: enums.SubCarrSpacing: No parameter help available
- Channel_Bw: enums.ChannelBwidth: No parameter help available
- Remote_Correct: bool: No parameter help available

get() → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:BWConfig:INVoke
value: GetStruct = driver.configure.nrSubMeas.bwConfig.invoke.get()
```

No command help available

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

set(*sub_carr_spacing: SubCarrSpacing*, *channel_bw: ChannelBwidth*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:BWConfig:INVoke
driver.configure.nrSubMeas.bwConfig.invoke.set(sub_carr_spacing = enums.
↪SubCarrSpacing.S15K, channel_bw = enums.ChannelBwidth.B005)
```

No command help available

param sub_carr_spacing
No help available

param channel_bw
No help available

6.1.1.2 Caggregation

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:CAGGregation:MCARrier
CONFigure:NRSub:MEASurement<Instance>:CAGGregation:DCARrier
```

class CaggregationCls

Caggregation commands group definition. 7 total commands, 3 Subgroups, 2 group commands

get_dcarrier() → CarrierComponent

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CAGGregation:DCARrier
value: enums.CarrierComponent = driver.configure.nrSubMeas.caggregation.get_
↪dcarrier()
```

No command help available

return
display_carrier: No help available

get_mcarrier() → CarrierComponent

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CAGGregation:MCARrier
value: enums.CarrierComponent = driver.configure.nrSubMeas.caggregation.get_
↪mcarrier()
```

Selects a component carrier for synchronization and for single carrier measurements.

return
meas_carrier: No help available

set_dcarrier(*display_carrier: CarrierComponent*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CAGGregation:DCARrier
driver.configure.nrSubMeas.caggregation.set_dcarrier(display_carrier = enums.
↪CarrierComponent.CC1)
```

No command help available

param display_carrier
No help available

set_mcarrier(*meas_carrier: CarrierComponent*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CAGGregation:MCARrier
driver.configure.nrSubMeas.caggregation.set_mcarrier(meas_carrier = enums.
↪CarrierComponent.CC1)
```

Selects a component carrier for synchronization and for single carrier measurements.

param meas_carrier
No help available

Cloning the Group

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

Subgroups

6.1.1.2.1 AcSpacing

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:CAGGregation:ACSPacing
```

class AcSpacingCls

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

set() → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CAGGregation:ACSPacing
driver.configure.nrSubMeas.caggregation.acSpacing.set()
```

Adjusts the component carrier frequencies, so that the carriers are aggregated contiguously, with nominal channel spacing.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CAGGregation:ACSPacing
driver.configure.nrSubMeas.caggregation.acSpacing.set_with_opc()
```

Adjusts the component carrier frequencies, so that the carriers are aggregated contiguously, with nominal channel spacing.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_NrFr1Meas.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.1.1.2.2 Cbandwidth

SCPI Command :

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

class CbandwidthCls

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

get_aggregated() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CAGGregation:CBANdwidth:AGGRegated
value: float = driver.configure.nrSubMeas.caggregation.cbandwidth.get_
↪aggregated()
```

Queries the width of the aggregated channel bandwidth.

return
ch_bandwidth: No help available

6.1.1.2.3 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.nrSubMeas.caggregation.frequency.clone()
```

Subgroups

6.1.1.2.3.1 Aggregated

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:LOW
CONFigure:NRSub:MEASurement<Instance>:CAGGregation:FREQuency:AGGRegated:CENTer
CONFigure:NRSub: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:NRSub:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:CENTer
value: float = driver.configure.nrSubMeas.caggregation.frequency.aggregated.get_
↪center()
```

Queries the center frequency of the aggregated bandwidth.

return
frequency_center: No help available

get_high() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:HIGH
value: float = driver.configure.nrSubMeas.caggregation.frequency.aggregated.get_
↪high()
```

Queries the upper edge of the aggregated bandwidth.

return
frequency_high: No help available

get_low() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:CAGGregation:FREQuency:AGGRegated:LOW
value: float = driver.configure.nrSubMeas.caggregation.frequency.aggregated.get_
↪low()
```

Queries the lower edge of the aggregated bandwidth.

return
frequency_low: No help available

6.1.1.3 Cc

class CcCls

Cc commands group definition. 39 total commands, 10 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.3.1 Allocation<Allocation>

RepCap Settings

```
# Range: Nr1 .. Nr1
rc = driver.configure.nrSubMeas.cc.allocation.repcap_allocation_get()
driver.configure.nrSubMeas.cc.allocation.repcap_allocation_set(repcap.Allocation.Nr1)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
```

class AllocationCls

Allocation commands group definition. 16 total commands, 2 Subgroups, 1 group commands Repeated Capability: Allocation, default value after init: Allocation.Nr1

class AllocationStruct

Response structure. Fields:

- Bandwidth_Part: enums.BandwidthPart: The current software version uses only one bandwidth part.
- Slot_Format: int: The current software version supports only slot format 1.
- Content: enums.ChannelTypeA: Type of channel to be measured
- Allocated_Slots: enums.AllocatedSlots: In the current software version, all UL slots must be scheduled and use the same allocation.

get(*carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → AllocationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
value: AllocationStruct = driver.configure.nrSubMeas.cc.allocation.
↪get(carrierComponent = repcap.CarrierComponent.Nr1, allocation = repcap.
↪Allocation.Default)
```

Selects the type of channel to be measured, for carrier <no>, allocation <a>. The other parameters in this command are fixed in the current software version.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

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

set(*bandwidth_part: BandwidthPart*, *slot_format: int*, *content: ChannelTypeA*, *allocated_slots: AllocatedSlots*, *carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
driver.configure.nrSubMeas.cc.allocation.set(bandwidth_part = enums.
↪BandwidthPart.BWP0, slot_format = 1, content = enums.ChannelTypeA.PUCCh,␣
↪allocated_slots = enums.AllocatedSlots.ALL, carrierComponent = repcap.
↪CarrierComponent.Nr1, allocation = repcap.Allocation.Default)
```

Selects the type of channel to be measured, for carrier <no>, allocation <a>. The other parameters in this command are fixed in the current software version.

param bandwidth_part
The current software version uses only one bandwidth part.

param slot_format
The current software version supports only slot format 1.

param content
Type of channel to be measured

param allocated_slots
In the current software version, all UL slots must be scheduled and use the same allocation.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

Cloning the Group

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

Subgroups

6.1.1.3.1.1 Pucch

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh
```

class PucchCls

Pucch commands group definition. 11 total commands, 7 Subgroups, 1 group commands

class PucchStruct

Response structure. Fields:

- Pucch_Format: enums.PucchFormat: PUCCH format
- No_Symbols: int: Number of allocated OFDM symbols in each uplink slot.
- Start_Symbol: int: Index of the first allocated symbol in each uplink slot.
- No_Rbs: int: Number of allocated UL RBs.

- Start_Rb: int: Index of the first allocated RB.

get(*carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → PucchStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh
value: PucchStruct = driver.configure.nrSubMeas.cc.allocation.pucch.
↪get(carrierComponent = repcap.CarrierComponent.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies settings related to the PUCCH allocation, for carrier <no>, allocation <a>. The ranges for the allocated RBs and symbols have dependencies, see 'RB allocation for uplink measurements' and 'Slots and symbols for PUSCH and PUCCH'.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

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

set(*pucch_format: PucchFormat*, *no_symbols: int*, *start_symbol: int*, *no_rbs: int*, *start_rb: int*, *carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh
driver.configure.nrSubMeas.cc.allocation.pucch.set(pucch_format = enums.
↪PucchFormat.F0, no_symbols = 1, start_symbol = 1, no_rbs = 1, start_rb = 1,␣
↪carrierComponent = repcap.CarrierComponent.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies settings related to the PUCCH allocation, for carrier <no>, allocation <a>. The ranges for the allocated RBs and symbols have dependencies, see 'RB allocation for uplink measurements' and 'Slots and symbols for PUSCH and PUCCH'.

param pucch_format
PUCCH format

param no_symbols
Number of allocated OFDM symbols in each uplink slot.

param start_symbol
Index of the first allocated symbol in each uplink slot.

param no_rbs
Number of allocated UL RBs.

param start_rb
Index of the first allocated RB.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

Cloning the Group

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

Subgroups

6.1.1.3.1.2 Dmrs

class DmrsCls

Dmrs 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.nrSubMeas.cc.allocation.pucch.dmrs.clone()
```

Subgroups

6.1.1.3.1.3 Did

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:DMRS:DID
```

class DidCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:DMRS:DID
value: int = driver.configure.nrSubMeas.cc.allocation.pucch.dmrs.did.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Specifies the DMRS ID, for carrier <no>, allocation <a>. See also method RsCMPX_NrFr1Meas.Configure.NrSubMeas.Cc. Allocation.Pucch.Dmrs.Init.set.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
idn: No help available

set(*idn: int*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:DMRS:DID
driver.configure.nrSubMeas.cc.allocation.pucch.dmrs.did.set(idn = 1,␣
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies the DMRS ID, for carrier <no>, allocation <a>. See also method RsCMPX_NrFr1Meas.Configure.NrSubMeas.Cc. Allocation.Pucch.Dmrs.Init.set.

param idn
No help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.4 Init

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:DMRS:INIT
```

class InitCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → DmrsInit

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:DMRS:INIT
value: enums.DmrsInit = driver.configure.nrSubMeas.cc.allocation.pucch.dmrs.
↪init.get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Specifies the type of ID used for initialization of the DMRS sequence generation for PUCCH format F2, for carrier <no>, allocation <a>.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
initialization: Cell ID or DMRS ID

set(*initialization: DmrsInit*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:DMRS:INIT
driver.configure.nrSubMeas.cc.allocation.pucch.dmrs.init.set(initialization =␣
↪enums.DmrsInit.CID, carrierComponentFour = repcap.CarrierComponentFour.Nr1,␣
↪allocation = repcap.Allocation.Default)
```

Specifies the type of ID used for initialization of the DMRS sequence generation for PUCCH format F2, for carrier <no>, allocation <a>.

param initialization
Cell ID or DMRS ID

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.5 Ghopping

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:GHOPping
```

class GhoppingCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → GroupHopping

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:GHOPping
value: enums.GroupHopping = driver.configure.nrSubMeas.cc.allocation.pucch.
↪ghopping.get(carrierComponentFour = repcap.CarrierComponentFour.Nr1,␣
↪allocation = repcap.Allocation.Default)
```

Specifies whether group hopping and/or sequence hopping are used for the PUCCH DMRS, for carrier <no>, allocation <a>.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
group_hopping: NEITher: no group hopping and no sequence hopping ENABle: group hopping DISable: sequence hopping

set(*group_hopping: GroupHopping*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:GHOPping
driver.configure.nrSubMeas.cc.allocation.pucch.ghopping.set(group_hopping =␣
↪enums.GroupHopping.DISable, carrierComponentFour = repcap.
↪CarrierComponentFour.Nr1, allocation = repcap.Allocation.Default)
```

Specifies whether group hopping and/or sequence hopping are used for the PUCCH DMRS, for carrier <no>, allocation <a>.

param group_hopping
NEITher: no group hopping and no sequence hopping ENABle: group hopping DISable: sequence hopping

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

Cloning the Group

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

Subgroups

6.1.1.3.1.6 Hid

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:GHOPping:HID
```

class HidCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:GHOPping:HID
value: int = driver.configure.nrSubMeas.cc.allocation.pucch.ghopping.hid.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Specifies the hopping ID, for carrier <no>, allocation <a>. See also method RsCMPX_NrFr1Meas.Configure.NrSubMeas.Cc. Allocation.Pucch.Ghopping.Init.set.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
idn: No help available

set(*idn: int*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:GHOPping:HID
driver.configure.nrSubMeas.cc.allocation.pucch.ghopping.hid.set(idn = 1,␣
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies the hopping ID, for carrier <no>, allocation <a>. See also method RsCMPX_NrFr1Meas.Configure.NrSubMeas.Cc. Allocation.Pucch.Ghopping.Init.set.

param idn
No help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.7 Init

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:GHOPping:INIT
```

class InitCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → GhopingInit

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:GHOPping:INIT
value: enums.GhopingInit = driver.configure.nrSubMeas.cc.allocation.pucch.
↪ghopping.init.get(carrierComponentFour = repcap.CarrierComponentFour.Nr1,␣
↪allocation = repcap.Allocation.Default)
```

Specifies the type of ID used to initialize group hopping and sequence hopping, for carrier <no>, allocation <a>.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
initialization: Cell ID or hopping ID

set(*initialization: GhopingInit*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:GHOPping:INIT
driver.configure.nrSubMeas.cc.allocation.pucch.ghopping.init.set(initialization␣
↪= enums.GhopingInit.CID, carrierComponentFour = repcap.CarrierComponentFour.
↪Nr1, allocation = repcap.Allocation.Default)
```

Specifies the type of ID used to initialize group hopping and sequence hopping, for carrier <no>, allocation <a>.

param initialization
Cell ID or hopping ID

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.8 IcShift

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:ICSHift
```

class IcShiftCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:ICSHift
value: int = driver.configure.nrSubMeas.cc.allocation.pucch.icShift.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Specifies the initial cyclic shift, for carrier <no>, allocation <a>.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
value: No help available

set(*value: int*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:ICSHift
driver.configure.nrSubMeas.cc.allocation.pucch.icShift.set(value = 1,␣
```

(continues on next page)

(continued from previous page)

```
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies the initial cyclic shift, for carrier <no>, allocation <a>.

param value
No help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.9 IsfHopping

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:ISFHopping
```

class IsfHoppingCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:ISFHopping
value: bool = driver.configure.nrSubMeas.cc.allocation.pucch.isfHopping.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

No command help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
enable: No help available

set(*enable: bool*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:ISFHopping
driver.configure.nrSubMeas.cc.allocation.pucch.isfHopping.set(enable = False,␣
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

No command help available

param enable
No help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.10 Occ

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:OCC
```

class OccCls

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

class OccStruct

Response structure. Fields:

- Length: int: OCC length
- Index: int: OCC index

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → OccStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:OCC
value: OccStruct = driver.configure.nrSubMeas.cc.allocation.pucch.occ.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Specifies the OCC length and index for PUCCH format F4, for carrier <no>, allocation <a>.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

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

set(*length: int*, *index: int*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:OCC
driver.configure.nrSubMeas.cc.allocation.pucch.occ.set(length = 1, index = 1,␣
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies the OCC length and index for PUCCH format F4, for carrier <no>, allocation <a>.

param length
OCC length

param index
OCC index

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.11 ShPrb

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:SHPRb
```

class ShPrbCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:SHPRb
value: int = driver.configure.nrSubMeas.cc.allocation.pucch.shPrb.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

No command help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
value: No help available

set(*value: int*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:SHPRb
driver.configure.nrSubMeas.cc.allocation.pucch.shPrb.set(value = 1,␣
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

No command help available

param value
No help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.12 TdoIndex

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUCCh:TDOindex
```

class TdoIndexCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:TDOindex
value: int = driver.configure.nrSubMeas.cc.allocation.pucch.tdoIndex.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Specifies the time domain OCC index for PUCCH format F1, for carrier <no>, allocation <a>.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
value: No help available

set(*value: int*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUCCh:TDOindex
driver.configure.nrSubMeas.cc.allocation.pucch.tdoIndex.set(value = 1,␣
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies the time domain OCC index for PUCCH format F1, for carrier <no>, allocation <a>.

param value
No help available

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.13 Pusch

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUSCh
```

class PuschCls

Pusch commands group definition. 4 total commands, 3 Subgroups, 1 group commands

class PuschStruct

Structure for setting input parameters. Fields:

- Mapping_Type: enums.MappingType: PUSCH mapping type
- No_Symbols: int: Number of allocated OFDM symbols in each uplink slot.
- Start_Symbol: int: Index of the first allocated symbol in each uplink slot. For mapping type A, only 0 is allowed.
- Auto: bool: Automatic detection of NoRBs and StartRB
- No_Rbs: int: Number of allocated UL RBs.
- Start_Rb: int: Index of the first allocated RB.
- Mod_Scheme: enums.ModulationScheme: Modulation scheme AUTO: Auto-detection BPSK, BPWS: /2-BPSK, /2-BPSK with shaping QPSK, Q16, Q64, Q256: QPSK, 16QAM, 64QAM, 256QAM

get(*carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → PuschStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh
value: PuschStruct = driver.configure.nrSubMeas.cc.allocation.pusch.
↪get(carrierComponent = repcap.CarrierComponent.Nr1, allocation = repcap.
↪Allocation.Default)
```

Specifies settings related to the PUSCH allocation, for carrier <no>, allocation <a>. The ranges for the allocated RBs and symbols have dependencies, see 'RB allocation for uplink measurements' and 'Slots and symbols for PUSCH and PUCCH'.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

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

set(*structure: PuschStruct*, *carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh
structure = driver.configure.nrSubMeas.cc.allocation.pusch.PuschStruct()
structure.Mapping_Type: enums.MappingType = enums.MappingType.A
```

(continues on next page)

(continued from previous page)

```
structure.No_Symbols: int = 1
structure.Start_Symbol: int = 1
structure.Auto: bool = False
structure.No_Rbs: int = 1
structure.Start_Rb: int = 1
structure.Mod_Scheme: enums.ModulationScheme = enums.ModulationScheme.AUTO
driver.configure.nrSubMeas.cc.allocation.pusch.set(structure, carrierComponent␣
↪= repcap.CarrierComponent.Nr1, allocation = repcap.Allocation.Default)
```

Specifies settings related to the PUSCH allocation, for carrier <no>, allocation <a>. The ranges for the allocated RBs and symbols have dependencies, see 'RB allocation for uplink measurements' and 'Slots and symbols for PUSCH and PUCCH'.

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

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

Cloning the Group

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

Subgroups

6.1.1.3.1.14 Additional

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUSCh:ADDitional
```

class AdditionalCls

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

class AdditionalStruct

Response structure. Fields:

- Dmrs_Length: int: Length of the DM-RS in symbols. The maximum value is limited by the 'maxLength' setting for the bandwidth part.
- Antenna_Port: int: Antenna port of the DM-RS, for transmission layer 1.
- Cdm_Groups: int: Number of DM-RS CDM groups without data.
- Antenna_Port_2: int: Antenna port of the DM-RS, for transmission layer 2.

get(*carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → AdditionalStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh:ADDitional
value: AdditionalStruct = driver.configure.nrSubMeas.cc.allocation.pusch.
↪additional.get(carrierComponent = repcap.CarrierComponent.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Configures special PUSCH settings, for carrier <no>, allocation <a>.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

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

set(*dmrs_length: int*, *antenna_port: int = None*, *cdm_groups: int = None*, *antenna_port_2: int = None*, *carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh:ADDitional
driver.configure.nrSubMeas.cc.allocation.pusch.additional.set(dmrs_length = 1,␣
↪antenna_port = 1, cdm_groups = 1, antenna_port_2 = 1, carrierComponent =␣
↪repcap.CarrierComponent.Nr1, allocation = repcap.Allocation.Default)
```

Configures special PUSCH settings, for carrier <no>, allocation <a>.

param dmrs_length
Length of the DM-RS in symbols. The maximum value is limited by the 'maxLength' setting for the bandwidth part.

param antenna_port
Antenna port of the DM-RS, for transmission layer 1.

param cdm_groups
Number of DM-RS CDM groups without data.

param antenna_port_2
Antenna port of the DM-RS, for transmission layer 2.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.15 Nlayers

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUSCh:NLAYers
```

class NlayersCls

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

get(*carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh:NLAYers
value: int = driver.configure.nrSubMeas.cc.allocation.pusch.nlayers.
↪get(carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Selects the number of layers transmitted by the UE, for carrier <no>, allocation <a>.

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

return
number: Number of layers

set(*number: int*, *carrierComponentFour=CarrierComponentFour.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh:NLAYers
driver.configure.nrSubMeas.cc.allocation.pusch.nlayers.set(number = 1,␣
↪carrierComponentFour = repcap.CarrierComponentFour.Nr1, allocation = repcap.
↪Allocation.Default)
```

Selects the number of layers transmitted by the UE, for carrier <no>, allocation <a>.

param number
Number of layers

param carrierComponentFour
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.1.16 Sgeneration

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>:PUSCh:SGENeration
```

class SgenerationCls

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

class SgenerationStruct

Response structure. Fields:

- Initialization: enums.Generator: PHY: physical cell ID used DID: DMRS ID used
- Dmrs_Id: int: ID for Initialization = DID.
- Nscid: int: Parameter nSCID.

get(*carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → SgenerationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh:SGENeration
value: SgenerationStruct = driver.configure.nrSubMeas.cc.allocation.pusch.
↪sgeneration.get(carrierComponent = repcap.CarrierComponent.Nr1, allocation =␣
↪repcap.Allocation.Default)
```

Configures the initialization of the DM-RS sequence generation, for carrier <no>, allocation <a>.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

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

set(*initialization: Generator*, *dmrs_id: int*, *nscid: int*, *carrierComponent=CarrierComponent.Nr1*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:ALLocation<Allocation>
↪:PUSCh:SGENeration
driver.configure.nrSubMeas.cc.allocation.pusch.sgeneration.set(initialization =␣
↪enums.Generator.DID, dmrs_id = 1, nscid = 1, carrierComponent = repcap.
↪CarrierComponent.Nr1, allocation = repcap.Allocation.Default)
```

Configures the initialization of the DM-RS sequence generation, for carrier <no>, allocation <a>.

param initialization
PHY: physical cell ID used DID: DMRS ID used

param dmrs_id
ID for Initialization = DID.

param nscid
Parameter nSCID.

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

6.1.1.3.2 BwPart

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart
```

class BwPartCls

BwPart commands group definition. 6 total commands, 2 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Sub_Car_Spacing: enums.SubCarrSpacing: Subcarrier spacing 15 kHz, 30 kHz, 60 kHz.
- Cyclic_Prefix: enums.CyclicPrefix: EXTended cyclic prefix is only possible for 60-kHz SC spacing.
- Number_Rb: int: Number of RBs in the bandwidth part.
- Start_Rb: int: Index of the first RB in the bandwidth part.

get(*bwp: BandwidthPart*, *carrierComponent=CarrierComponent.Nr1*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart
value: GetStruct = driver.configure.nrSubMeas.cc.bwPart.get(bwp = enums.
↪BandwidthPart.BWP0, carrierComponent = repcap.CarrierComponent.Nr1)
```

Configures basic properties of the <BWP> on carrier <no>. For dependencies of the RB ranges, see 'Resource elements, grids and blocks'.

param bwp
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

set(*bwp: BandwidthPart*, *sub_car_spacing: SubCarrSpacing*, *cyclic_prefix: CyclicPrefix*, *number_rb: int*, *start_rb: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart
driver.configure.nrSubMeas.cc.bwPart.set(bwp = enums.BandwidthPart.BWP0, sub_
↪car_spacing = enums.SubCarrSpacing.S15K, cyclic_prefix = enums.CyclicPrefix.
↪EXTended, number_rb = 1, start_rb = 1, carrierComponent = repcap.
↪CarrierComponent.Nr1)
```

Configures basic properties of the <BWP> on carrier <no>. For dependencies of the RB ranges, see 'Resource elements, grids and blocks'.

param bwp
No help available

param sub_car_spacing
Subcarrier spacing 15 kHz, 30 kHz, 60 kHz.

param cyclic_prefix
EXTended cyclic prefix is only possible for 60-kHz SC spacing.

param number_rb
Number of RBs in the bandwidth part.

param start_rb
Index of the first RB in the bandwidth part.

param carrierComponent
optional repeated capability selector. Default value: Nr1

Cloning the Group

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

Subgroups

6.1.1.3.2.1 Pucch

class PucchCls

Pucch 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.nrSubMeas.cc.bwPart.pucch.clone()
```

Subgroups

6.1.1.3.2.2 AdMrs

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUCCh:ADMRs
```

class AdMrsCls

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

get(*bwp: BandwidthPart*, *carrierComponent=CarrierComponent.Nr1*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUCCh:ADMRs
value: bool = driver.configure.nrSubMeas.cc.bwPart.pucch.adMrs.get(bwp = enums.
↪BandwidthPart.BWP0, carrierComponent = repcap.CarrierComponent.Nr1)
```

Specifies whether the PUCCH in the <BWP> on carrier <no> uses an additional DMRS.

param bwp
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
additional_dmrs: No help available

set(*bwp: BandwidthPart*, *additional_dmrs: bool*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUCCh:ADMRs
driver.configure.nrSubMeas.cc.bwPart.pucch.adMrs.set(bwp = enums.BandwidthPart.
↪BWP0, additional_dmrs = False, carrierComponent = repcap.CarrierComponent.Nr1)
```

Specifies whether the PUCCH in the <BWP> on carrier <no> uses an additional DMRS.

param bwp
No help available

param additional_dmrs
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.2.3 Phbpsk

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUCCh:PHBPsk
```

class PhbpskCls

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

get(*bwp: BandwidthPart*, *carrierComponent=CarrierComponent.Nr1*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUCCh:PHBPsk
value: bool = driver.configure.nrSubMeas.cc.bwPart.pucch.phbpsk.get(bwp = enums.
↪BandwidthPart.BWP0, carrierComponent = repcap.CarrierComponent.Nr1)
```

Specifies whether the PUCCH in the <BWP> on carrier <no> uses /2-BPSK modulation.

param bwp
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
pi_half_pbsk: No help available

set(*bwp: BandwidthPart*, *pi_half_pbsk: bool*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUCCh:PHBPsk
driver.configure.nrSubMeas.cc.bwPart.pucch.phbpsk.set(bwp = enums.BandwidthPart.
↪BWP0, pi_half_pbsk = False, carrierComponent = repcap.CarrierComponent.Nr1)
```

Specifies whether the PUCCH in the <BWP> on carrier <no> uses /2-BPSK modulation.

param bwp
No help available

param pi_half_pbsk
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.2.4 Pusch

class PuschCls

Pusch 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.configure.nrSubMeas.cc.bwPart.pusch.clone()
```

Subgroups

6.1.1.3.2.5 DftPrecoding

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DFTPrecoding
```

class DftPrecodingCls

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

get(*bwp: BandwidthPart*, *carrierComponent=CarrierComponent.Nr1*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DFTPrecoding
value: bool = driver.configure.nrSubMeas.cc.bwPart.pusch.dftPrecoding.get(bwp =␣
↪enums.BandwidthPart.BWP0, carrierComponent = repcap.CarrierComponent.Nr1)
```

Specifies whether the <BWP> on carrier <no> uses a transform precoding function.

param bwp
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
dft_precoding: OFF: No transform precoding. ON: With transform precoding.

set(*bwp: BandwidthPart*, *dft_precoding: bool*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DFTPrecoding
driver.configure.nrSubMeas.cc.bwPart.pusch.dftPrecoding.set(bwp = enums.
↪BandwidthPart.BWP0, dft_precoding = False, carrierComponent = repcap.
↪CarrierComponent.Nr1)
```

Specifies whether the <BWP> on carrier <no> uses a transform precoding function.

param bwp
No help available

param dft_precoding
OFF: No transform precoding. ON: With transform precoding.

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.2.6 Dmta

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DMTA
```

class DmtaCls

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

class GetStruct

Response structure. Fields:

- Config_Type: int: DM-RS setting 'dmrs-Type'.
- Add_Position: int: DM-RS setting 'dmrs-AdditionalPosition'.
- Max_Length: int: DM-RS setting 'maxLength'.

get(*bwp: BandwidthPart*, *carrierComponent=CarrierComponent.Nr1*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DMTA
value: GetStruct = driver.configure.nrSubMeas.cc.bwPart.pusch.dmta.get(bwp =␣
↪enums.BandwidthPart.BWP0, carrierComponent = repcap.CarrierComponent.Nr1)
```

Configures the DM-RS for mapping type A. The settings apply to the <BWP> on carrier <no>.

param bwp
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

set(*bwp: BandwidthPart*, *config_type: int*, *add_position: int*, *max_length: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DMTA
driver.configure.nrSubMeas.cc.bwPart.pusch.dmta.set(bwp = enums.BandwidthPart.
↪BWP0, config_type = 1, add_position = 1, max_length = 1, carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

Configures the DM-RS for mapping type A. The settings apply to the <BWP> on carrier <no>.

param bwp
No help available

param config_type
DM-RS setting 'dmrs-Type'.

param add_position
DM-RS setting 'dmrs-AdditionalPosition'.

param max_length
DM-RS setting 'maxLength'.

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.2.7 Dmtb

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DMTB
```

class DmtbCls

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

class GetStruct

Response structure. Fields:

- Config_Type: int: DM-RS setting 'dmrs-Type'.
- Add_Position: int: DM-RS setting 'dmrs-AdditionalPosition'.
- Max_Length: int: DM-RS setting 'maxLength'.

get(*bwp: BandwidthPart*, *carrierComponent=CarrierComponent.Nr1*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DMTB
value: GetStruct = driver.configure.nrSubMeas.cc.bwPart.pusch.dmtb.get(bwp =␣
↪enums.BandwidthPart.BWP0, carrierComponent = repcap.CarrierComponent.Nr1)
```

Configures the DM-RS for mapping type B. The settings apply to the <BWP> on carrier <no>.

param bwp
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

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

set(*bwp: BandwidthPart*, *config_type: int*, *add_position: int*, *max_length: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:BWPart:PUSCh:DMTB
driver.configure.nrSubMeas.cc.bwPart.pusch.dmtb.set(bwp = enums.BandwidthPart.
↪BWP0, config_type = 1, add_position = 1, max_length = 1, carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

Configures the DM-RS for mapping type B. The settings apply to the <BWP> on carrier <no>.

param bwp
No help available

param config_type
DM-RS setting 'dmrs-Type'.

param add_position
DM-RS setting 'dmrs-AdditionalPosition'.

param max_length
DM-RS setting 'maxLength'.

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.3 Cbandwidth

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:CBANdwidth
```

class CbandwidthCls

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

get(*carrierComponent=CarrierComponent.Nr1*) → ChannelBwidth

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:CBANdwidth
value: enums.ChannelBwidth = driver.configure.nrSubMeas.cc.cbandwidth.
↪get(carrierComponent = repcap.CarrierComponent.Nr1)
```

Specifies the channel bandwidth of carrier <no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
channel_bw: Channel bandwidth 5 MHz to 100 MHz (Bxxx = xxx MHz) .

set(*channel_bw: ChannelBwidth*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:CBANdwidth
driver.configure.nrSubMeas.cc.cbandwidth.set(channel_bw = enums.ChannelBwidth.
↪B005, carrierComponent = repcap.CarrierComponent.Nr1)
```

Specifies the channel bandwidth of carrier <no>.

param channel_bw
Channel bandwidth 5 MHz to 100 MHz (Bxxx = xxx MHz) .

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.4 Frequency

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:FREQuency
```

class FrequencyCls

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

get(*carrierComponent=CarrierComponent.Nr1*) → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:FREQuency
value: float = driver.configure.nrSubMeas.cc.frequency.get(carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

Selects the center frequency of carrier <no>. Without carrier aggregation, you can omit CC<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'. For the supported frequency range, see 'Frequency ranges'.

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
analyzer_freq: No help available

set(*analyzer_freq: float*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:FREQuency
driver.configure.nrSubMeas.cc.frequency.set(analyzer_freq = 1.0,␣
↪carrierComponent = repcap.CarrierComponent.Nr1)
```

Selects the center frequency of carrier <no>. Without carrier aggregation, you can omit CC<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'. For the supported frequency range, see 'Frequency ranges'.

param analyzer_freq
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.5 Nallocations

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:NALLocations
```

class NallocationsCls

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

get(*carrierComponent=CarrierComponent.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:NALLocations
value: int = driver.configure.nrSubMeas.cc.nallocations.get(carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

Number of allocations to be configured, for carrier <no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
number: For the measured carrier, only 1 is allowed.

set(*number: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:NALLocations
driver.configure.nrSubMeas.cc.nallocations.set(number = 1, carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

Number of allocations to be configured, for carrier <no>.

param number
For the measured carrier, only 1 is allowed.

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.6 NbwParts

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:NBWParts
```

class NbwPartsCls

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

get(*carrierComponent=CarrierComponent.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:NBWParts
value: int = driver.configure.nrSubMeas.cc.nbwParts.get(carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

No command help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
no_of_bw_parts: No help available

set(*no_of_bw_parts: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:NBWParts
driver.configure.nrSubMeas.cc.nbwParts.set(no_of_bw_parts = 1, carrierComponent␣
↪= repcap.CarrierComponent.Nr1)
```

No command help available

param no_of_bw_parts
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.7 PlcId

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:PLCid
```

class PlcIdCls

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

get(*carrierComponent=CarrierComponent.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:PLCid
value: int = driver.configure.nrSubMeas.cc.plcId.get(carrierComponent = repcap.
↪CarrierComponent.Nr1)
```

Specifies the physical cell ID of carrier <no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
physical_cell_id: No help available

set(*physical_cell_id: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:PLCid
driver.configure.nrSubMeas.cc.plcId.set(physical_cell_id = 1, carrierComponent␣
↪= repcap.CarrierComponent.Nr1)
```

Specifies the physical cell ID of carrier <no>.

param physical_cell_id
No help available

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.8 Rpool

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol
```

class RpoolCls

Rpool commands group definition. 10 total commands, 2 Subgroups, 1 group commands

class RpoolStruct

Response structure. Fields:

- No_Rbs: int: No parameter help available
- Start_Rb: int: No parameter help available
- Sub_Chan_Size: enums.SubChanSize: No parameter help available
- No_Sub_Chans: int: No parameter help available

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → RpoolStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol
value: RpoolStruct = driver.configure.nrSubMeas.cc.rpool.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

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

set(*no_rbs: int*, *start_rb: int*, *sub_chan_size: SubChanSize*, *no_sub_chans: int*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol
driver.configure.nrSubMeas.cc.rpool.set(no_rbs = 1, start_rb = 1, sub_chan_size␣
↪= enums.SubChanSize.RB10, no_sub_chans = 1, carrierComponentOne = repcap.
↪CarrierComponentOne.Nr1)
```

No command help available

param no_rbs
No help available

param start_rb
No help available

param sub_chan_size
No help available

param no_sub_chans
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

Cloning the Group

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

Subgroups

6.1.1.3.8.1 Pscch

class PscchCls

Pscch 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.configure.nrSubMeas.cc.rpool.pscch.clone()
```

Subgroups

6.1.1.3.8.2 Did

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:DID
```

class DidCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:DID
value: int = driver.configure.nrSubMeas.cc.rpool.pscch.did.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
dmrs_id: No help available

set(*dmrs_id: int*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:DID
driver.configure.nrSubMeas.cc.rpool.pscch.did.set(dmrs_id = 1,␣
↪carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param dmrs_id
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.3 Nrb

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:NRB
```

class NrbCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → SubChanSize

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:NRB
value: enums.SubChanSize = driver.configure.nrSubMeas.cc.rpool.pscch.nrb.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
no_rbs: No help available

set(*no_rbs: SubChanSize*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:NRB
driver.configure.nrSubMeas.cc.rpool.pscch.nrb.set(no_rbs = enums.SubChanSize.
↪RB10, carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param no_rbs
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.4 Nsymbols

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:NSYMbols
```

class NsymbolsCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:NSYMbols
value: int = driver.configure.nrSubMeas.cc.rpool.pscch.nsymbols.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
no_symbols: No help available

set(*no_symbols: int*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSCCh:NSYMbols
driver.configure.nrSubMeas.cc.rpool.pscch.nsymbols.set(no_symbols = 1,␣
↪carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param no_symbols
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.5 Pssch

class PsschCls

Pssch commands group definition. 6 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.3.8.6 Dport

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:DPORt
```

class DportCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → DmrsPort

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:DPORt
value: enums.DmrsPort = driver.configure.nrSubMeas.cc.rpool.pssch.dport.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
dmrs_port: No help available

set(*dmrs_port: DmrsPort*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:DPORt
driver.configure.nrSubMeas.cc.rpool.pssch.dport.set(dmrs_port = enums.DmrsPort.
↪ALL, carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param dmrs_port
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.7 Mscheme

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:MSCHeme
```

class MschemeCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → ModulationSchemeB

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:MSCHeme
value: enums.ModulationSchemeB = driver.configure.nrSubMeas.cc.rpool.pssch.
↪mscheme.get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
mod_scheme: No help available

set(*mod_scheme: ModulationSchemeB*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:MSCHeme
driver.configure.nrSubMeas.cc.rpool.pssch.mscheme.set(mod_scheme = enums.
↪ModulationSchemeB.Q16, carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param mod_scheme
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.8 Ndmrs

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NDMRs
```

class NdmrsCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NDMRs
value: int = driver.configure.nrSubMeas.cc.rpool.pssch.ndmrs.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
no_dmrs: No help available

set(*no_dmrs: int*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NDMRs
driver.configure.nrSubMeas.cc.rpool.pssch.ndmrs.set(no_dmrs = 1,␣
↪carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param no_dmrs
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.9 Nlayers

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NLAYers
```

class NlayersCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NLAYers
value: int = driver.configure.nrSubMeas.cc.rpool.pssch.nlayers.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
no_layers: No help available

set(*no_layers: int*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NLAYers
driver.configure.nrSubMeas.cc.rpool.pssch.nlayers.set(no_layers = 1,␣
↪carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param no_layers
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.10 NsubChannels

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NSUBchannels
```

class NsubChannelsCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NSUBchannels
value: int = driver.configure.nrSubMeas.cc.rpool.pssch.nsubChannels.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
no_subchannels: No help available

set(*no_subchannels: int*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NSUBchannels
driver.configure.nrSubMeas.cc.rpool.pssch.nsubChannels.set(no_subchannels = 1,␣
↪carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param no_subchannels
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.8.11 Nsymbols

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NSYMbols
```

class NsymbolsCls

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

get(*carrierComponentOne=CarrierComponentOne.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NSYMbols
value: int = driver.configure.nrSubMeas.cc.rpool.pssch.nsymbols.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

return
no_symbols: No help available

set(*no_symbols: int*, *carrierComponentOne=CarrierComponentOne.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:RPOol:PSSCh:NSYMbols
driver.configure.nrSubMeas.cc.rpool.pssch.nsymbols.set(no_symbols = 1,␣
↪carrierComponentOne = repcap.CarrierComponentOne.Nr1)
```

No command help available

param no_symbols
No help available

param carrierComponentOne
optional repeated capability selector. Default value: Nr1

6.1.1.3.9 TaPosition

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:TAPosition
```

class TaPositionCls

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

get(*carrierComponent=CarrierComponent.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:TAPosition
value: int = driver.configure.nrSubMeas.cc.taPosition.get(carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

Specifies the 'dmrs-TypeA-Position' for carrier <no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
position: Number of the first DM-RS symbol for mapping type A.

set(*position: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:TAPosition
driver.configure.nrSubMeas.cc.taPosition.set(position = 1, carrierComponent =␣
↪repcap.CarrierComponent.Nr1)
```

Specifies the 'dmrs-TypeA-Position' for carrier <no>.

param position
Number of the first DM-RS symbol for mapping type A.

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.3.10 TxBwidth

class TxBwidthCls

TxBwidth 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.nrSubMeas.cc.txBwidth.clone()
```

Subgroups

6.1.1.3.10.1 Offset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:TXBWidth:OFFSet
```

class OffsetCls

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

get(*carrierComponent=CarrierComponent.Nr1*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:TXBWidth:OFFSet
value: int = driver.configure.nrSubMeas.cc.txBwidth.offset.get(carrierComponent␣
↪= repcap.CarrierComponent.Nr1)
```

Specifies the offset to carrier (TxBW offset) of carrier <no>.

param carrierComponent
optional repeated capability selector. Default value: Nr1

return
offset: Number of RBs

set(*offset: int*, *carrierComponent=CarrierComponent.Nr1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>[:CC<no>]:TXBWidth:OFFSet
driver.configure.nrSubMeas.cc.txBwidth.offset.set(offset = 1, carrierComponent␣
↪= repcap.CarrierComponent.Nr1)
```

Specifies the offset to carrier (TxBW offset) of carrier <no>.

param offset
Number of RBs

param carrierComponent
optional repeated capability selector. Default value: Nr1

6.1.1.4 Ccall

class CcallCls

Ccall 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.nrSubMeas.ccall.clone()
```

Subgroups

6.1.1.4.1 TxBwidth

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:CCALl:TXBWidth:SCSPacing
```

class TxBwidthCls

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

get_sc_spacing() → SubCarrSpacing

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CCALl:TXBWidth:SCSPacing
value: enums.SubCarrSpacing = driver.configure.nrSubMeas.ccall.txBwidth.get_sc_
↪spacing()
```

Selects the subcarrier spacing for all carriers.

return
used_scs: No help available

set_sc_spacing(*used_scs: SubCarrSpacing*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:CCALl:TXBWidth:SCSPacing
driver.configure.nrSubMeas.ccall.txBwidth.set_sc_spacing(used_scs = enums.
↪SubCarrSpacing.S15K)
```

Selects the subcarrier spacing for all carriers.

param used_scs
No help available

6.1.1.5 ListPy

class ListPyCls

ListPy commands group definition. 19 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.5.1 PlcId

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:PLCid:MODE
```

class PlcIdCls

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

get_mode() → ParameterSetMode

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:PLCid:MODE
value: enums.ParameterSetMode = driver.configure.nrSubMeas.listPy.plcId.get_
↪mode()
```

Selects which physical cell ID setting is used for list mode measurements.

return
mode: - GLOBal: The global setting is used for all segments, see CONFigure:NRSub:MEASi[:CCno]:PLCid. - LIST: The cell ID is configured per segment, see CONFigure:NRSub:MEASi:LIST:SEGMentno[:CCcc]:PLCid.

set_mode(*mode: ParameterSetMode*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:PLCid:MODE
driver.configure.nrSubMeas.listPy.plcId.set_mode(mode = enums.ParameterSetMode.
↪GLOBal)
```

Selects which physical cell ID setting is used for list mode measurements.

param mode

- GLOBal: The global setting is used for all segments, see CONFigure:NRSub:MEASi[:CCno]:PLCid.
- LIST: The cell ID is configured per segment, see CONFigure:NRSub:MEASi:LIST:SEGMentno[:CCcc]:PLCid.

6.1.1.5.2 Segment

RepCap Settings

```
# Range: Nr1 .. Nr512
rc = driver.configure.nrSubMeas.listPy.segment.repcap_sEGMent_get()
driver.configure.nrSubMeas.listPy.segment.repcap_sEGMent_set(repcap.SEGMent.Nr1)
```

class SegmentCls

Segment commands group definition. 18 total commands, 4 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.nrSubMeas.listPy.segment.clone()
```

Subgroups

6.1.1.5.2.1 Caggregation

class CaggregationCls

Caggregation 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.nrSubMeas.listPy.segment.caggregation.clone()
```

Subgroups

6.1.1.5.2.2 AcSpacing

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:CAGGregation:ACSPacing
```

class AcSpacingCls

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

set(*sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>
↪:CAGGregation:ACSPacing
driver.configure.nrSubMeas.listPy.segment.caggregation.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

6.1.1.5.2.3 Mcarrier

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:CAGGregation:MCARrier
```

class McarrierCls

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

get(*sEGMent=SEGMent.Default*) → CarrierComponent

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>
↪:CAGGregation:MCARrier
value: enums.CarrierComponent = driver.configure.nrSubMeas.listPy.segment.
↪caggregation.mcarrier.get(sEGMent = repcap.SEGMent.Default)
```

Selects a component carrier for synchronization and single carrier measurements.

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

return
meas_carrier: No help available

set(*meas_carrier: CarrierComponent*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>
↪:CAGGregation:MCARrier
driver.configure.nrSubMeas.listPy.segment.caggregation.mcarrier.set(meas_
↪carrier = enums.CarrierComponent.CC1, sEGMent = repcap.SEGMent.Default)
```

Selects a component carrier for synchronization and single carrier measurements.

param meas_carrier
No help available

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

6.1.1.5.2.4 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.nrSubMeas.listPy.segment.cc.repcap_carrierComponent_get()
driver.configure.nrSubMeas.listPy.segment.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 13 total commands, 8 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.nrSubMeas.listPy.segment.cc.clone()
```

Subgroups

6.1.1.5.2.5 Allocation<Allocation>

RepCap Settings

```
# Range: Nr1 .. Nr1
rc = driver.configure.nrSubMeas.listPy.segment.cc.allocation.repcap_allocation_get()
driver.configure.nrSubMeas.listPy.segment.cc.allocation.repcap_allocation_set(repcap.
↪Allocation.Nr1)
```

class AllocationCls

Allocation commands group definition. 3 total commands, 1 Subgroups, 0 group commands Repeated Capability: Allocation, default value after init: Allocation.Nr1

Cloning the Group

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

Subgroups

6.1.1.5.2.6 Pusch

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:ALLocation<Allocation>
↪:PUSCh
```

class PuschCls

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

class PuschStruct

Structure for setting input parameters. Fields:

- Mapping_Type: enums.MappingType: PUSCH mapping type
- No_Symbols: int: Number of allocated OFDM symbols in the measured slot.
- Start_Symbol: int: Index of the first allocated symbol in the measured slot. For mapping type A, only 0 is allowed.
- Nrb_Auto: bool: Automatic detection of NoRBs and StartRB
- No_Rb: int: Number of allocated RBs in the measured slot.
- Start_Rb: int: Index of the first allocated RB in the measured slot.
- Mod_Scheme: enums.ModulationScheme: Modulation scheme AUTO: Auto-detection BPSK, BPWS: /2-BPSK, /2-BPSK with shaping QPSK, Q16, Q64, Q256: QPSK, 16QAM, 64QAM, 256QAM

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*, *allocation=Allocation.Default*) → PuschStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:ALLocation<Allocation>:PUSCh
value: PuschStruct = driver.configure.nrSubMeas.listPy.segment.cc.allocation.
↪pusch.get(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default, allocation = repcap.Allocation.Default)
```

Specifies settings related to the PUSCH allocation, for carrier <cc>, allocation <a> in segment <no>. The ranges for the allocated RBs and symbols have dependencies, see 'RB allocation for uplink measurements' and 'Slots and symbols for PUSCH and PUCCH'.

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')

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

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

set(*structure: PuschStruct*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:ALLocation<Allocation>:PUSCh
structure = driver.configure.nrSubMeas.listPy.segment.cc.allocation.pusch.
↪PuschStruct()
structure.Mapping_Type: enums.MappingType = enums.MappingType.A
structure.No_Symbols: int = 1
structure.Start_Symbol: int = 1
structure.Nrb_Auto: bool = False
structure.No_Rb: int = 1
structure.Start_Rb: int = 1
structure.Mod_Scheme: enums.ModulationScheme = enums.ModulationScheme.AUTO
driver.configure.nrSubMeas.listPy.segment.cc.allocation.pusch.set(structure,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default, allocation = repcap.Allocation.Default)
```

Specifies settings related to the PUSCH allocation, for carrier <cc>, allocation <a> in segment <no>. The ranges for the allocated RBs and symbols have dependencies, see 'RB allocation for uplink measurements' and 'Slots and symbols for PUSCH and PUCCH'.

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

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')

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

Cloning the Group

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

Subgroups

6.1.1.5.2.7 Additional

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:ALLocation<Allocation>
↪:PUSCh:ADDitional
```

class AdditionalCls

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

class AdditionalStruct

Response structure. Fields:

- Dmrs_Length: int: Length of the DM-RS in symbols. The maximum value is limited by the 'maxLength' setting for the bandwidth part.
- Antenna_Port: int: Antenna port of the DM-RS.

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*, *allocation=Allocation.Default*) → AdditionalStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:ALLocation<Allocation>:PUSCh:ADDitional
value: AdditionalStruct = driver.configure.nrSubMeas.listPy.segment.cc.
↪allocation.pusch.additional.get(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default, allocation = repcap.
↪Allocation.Default)
```

Configures special PUSCH settings, for carrier <cc>, allocation <a> in segment <no>.

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')

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

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

set(*dmrs_length: int*, *antenna_port: int = None*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:ALLocation<Allocation>:PUSCh:ADDitional
driver.configure.nrSubMeas.listPy.segment.cc.allocation.pusch.additional.
↪set(dmrs_length = 1, antenna_port = 1, sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default, allocation = repcap.
↪Allocation.Default)
```

Configures special PUSCH settings, for carrier <cc>, allocation <a> in segment <no>.

param dmrs_length
Length of the DM-RS in symbols. The maximum value is limited by the 'maxLength' setting for the bandwidth part.

param antenna_port
Antenna port of the DM-RS.

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')

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

6.1.1.5.2.8 Sgeneration

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:ALLocation<Allocation>
↪:PUSCh:SGENeration
```

`class SgenerationCls`

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

`class SgenerationStruct`

Response structure. Fields:

- Initialization: enums.Generator: PHY: physical cell ID used DID: DMRS ID used
- Dmrs_Id: int: ID for Initialization = DID.
- Nscid: int: Parameter nSCID.

`get`(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*, *allocation=Allocation.Default*) → SgenerationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:ALLocation<Allocation>:PUSCh:SGENeration
value: SgenerationStruct = driver.configure.nrSubMeas.listPy.segment.cc.
↪allocation.pusch.sgeneration.get(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default, allocation = repcap.
↪Allocation.Default)
```

Configures the initialization of the DM-RS sequence generation, for carrier <cc>, allocation <a> in segment <no>.

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')

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

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

set(*initialization: Generator*, *dmrs_id: int*, *nscid: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*, *allocation=Allocation.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:ALLocation<Allocation>:PUSCh:SGENeration
driver.configure.nrSubMeas.listPy.segment.cc.allocation.pusch.sgeneration.
↪set(initialization = enums.Generator.DID, dmrs_id = 1, nscid = 1, sEGMent =␣
↪repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.Default,␣
↪allocation = repcap.Allocation.Default)
```

Configures the initialization of the DM-RS sequence generation, for carrier <cc>, allocation <a> in segment <no>.

param initialization
PHY: physical cell ID used DID: DMRS ID used

param dmrs_id
ID for Initialization = DID.

param nscid
Parameter nSCID.

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')

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

6.1.1.5.2.9 BwPart

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:BWPart
```

class BwPartCls

BwPart commands group definition. 4 total commands, 1 Subgroups, 1 group commands

class GetStruct

Response structure. Fields:

- Sub_Car_Spacing: enums.SubCarrSpacing: Subcarrier spacing 15 kHz, 30 kHz, 60 kHz.
- Cyclic_Prefix: enums.CyclicPrefix: EXTended cyclic prefix is only possible for 60-kHz SC spacing.
- Number_Rb: int: Number of RBs in the bandwidth part.
- Start_Rb: int: Index of the first RB in the bandwidth part.

get(*bwp: BandwidthPart*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:BWPart
value: GetStruct = driver.configure.nrSubMeas.listPy.segment.cc.bwPart.get(bwp␣
↪= enums.BandwidthPart.BWP0, sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Configures basic properties of the <BWP> on carrier <cc> in segment <no>. For dependencies of the RB ranges, see 'Resource elements, grids and blocks'.

param bwp
No help available

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 GetStruct structure arguments.

set(*bwp: BandwidthPart*, *sub_car_spacing: SubCarrSpacing*, *cyclic_prefix: CyclicPrefix*, *number_rb: int*, *start_rb: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:BWPart
driver.configure.nrSubMeas.listPy.segment.cc.bwPart.set(bwp = enums.
↪BandwidthPart.BWP0, sub_car_spacing = enums.SubCarrSpacing.S15K, cyclic_
↪prefix = enums.CyclicPrefix.EXTended, number_rb = 1, start_rb = 1, sEGMent =␣
↪repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.Default)
```

Configures basic properties of the <BWP> on carrier <cc> in segment <no>. For dependencies of the RB ranges, see 'Resource elements, grids and blocks'.

param bwp
No help available

param sub_car_spacing
Subcarrier spacing 15 kHz, 30 kHz, 60 kHz.

param cyclic_prefix
EXTended cyclic prefix is only possible for 60-kHz SC spacing.

param number_rb
Number of RBs in the bandwidth part.

param start_rb
Index of the first RB in the bandwidth part.

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.nrSubMeas.listPy.segment.cc.bwPart.clone()
```

Subgroups

6.1.1.5.2.10 Pusch

class PuschCls

Pusch 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.configure.nrSubMeas.listPy.segment.cc.bwPart.pusch.clone()
```

Subgroups

6.1.1.5.2.11 DftPrecoding

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:BWPart:PUSCh:DFTPrecoding
```

class DftPrecodingCls

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

get(*bwp: BandwidthPart*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:BWPart:PUSCh:DFTPrecoding
value: bool = driver.configure.nrSubMeas.listPy.segment.cc.bwPart.pusch.
↪dftPrecoding.get(bwp = enums.BandwidthPart.BWP0, sEGMent = repcap.SEGMent.
↪Default, carrierComponent = repcap.CarrierComponent.Default)
```

Specifies whether the <BWP> on carrier <cc> in segment <no> uses a transform precoding function.

param bwp
No help available

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
dft_precoding: OFF: No transform precoding. ON: With transform precoding.

set(*bwp: BandwidthPart*, *dft_precoding: bool*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:BWPart:PUSCh:DFTPrecoding
driver.configure.nrSubMeas.listPy.segment.cc.bwPart.pusch.dftPrecoding.set(bwp␣
↪= enums.BandwidthPart.BWP0, dft_precoding = False, sEGMent = repcap.SEGMent.
↪Default, carrierComponent = repcap.CarrierComponent.Default)
```

Specifies whether the <BWP> on carrier <cc> in segment <no> uses a transform precoding function.

param bwp
No help available

param dft_precoding
OFF: No transform precoding. ON: With transform precoding.

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')

6.1.1.5.2.12 Dmta

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:BWPart:PUSCh:DMTA
```

class DmtaCls

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

class GetStruct

Response structure. Fields:

- Config_Type: int: DM-RS setting 'dmrs-Type'.
- Add_Position: int: DM-RS setting 'dmrs-AdditionalPosition'.
- Max_Length: int: DM-RS setting 'maxLength'.

get(*bwp: BandwidthPart*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:BWPart:PUSCh:DMTA
value: GetStruct = driver.configure.nrSubMeas.listPy.segment.cc.bwPart.pusch.
↪dmta.get(bwp = enums.BandwidthPart.BWP0, sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Configures the DM-RS for mapping type A. The settings apply to the <BWP> on carrier <cc> in segment <no>.

param bwp
No help available

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 GetStruct structure arguments.

set(*bwp: BandwidthPart*, *config_type: int*, *add_position: int*, *max_length: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:BWPart:PUSCh:DMTA
driver.configure.nrSubMeas.listPy.segment.cc.bwPart.pusch.dmta.set(bwp = enums.
↪BandwidthPart.BWP0, config_type = 1, add_position = 1, max_length = 1,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Configures the DM-RS for mapping type A. The settings apply to the <BWP> on carrier <cc> in segment <no>.

param bwp
No help available

param config_type
DM-RS setting 'dmrs-Type'.

param add_position
DM-RS setting 'dmrs-AdditionalPosition'.

param max_length
DM-RS setting 'maxLength'.

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')

6.1.1.5.2.13 Dmtb

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:BWPart:PUSCh:DMTB
```

class DmtbCls

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

class GetStruct

Response structure. Fields:

- Config_Type: int: DM-RS setting 'dmrs-Type'.
- Add_Position: int: DM-RS setting 'dmrs-AdditionalPosition'.
- Max_Length: int: DM-RS setting 'maxLength'.

get(*bwp: BandwidthPart*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:BWPart:PUSCh:DMTB
value: GetStruct = driver.configure.nrSubMeas.listPy.segment.cc.bwPart.pusch.
↪dmtb.get(bwp = enums.BandwidthPart.BWP0, sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Configures the DM-RS for mapping type B. The settings apply to the <BWP> on carrier <cc> in segment <no>.

param bwp
No help available

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 GetStruct structure arguments.

set(*bwp: BandwidthPart*, *config_type: int*, *add_position: int*, *max_length: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:BWPart:PUSCh:DMTB
driver.configure.nrSubMeas.listPy.segment.cc.bwPart.pusch.dmtb.set(bwp = enums.
↪BandwidthPart.BWP0, config_type = 1, add_position = 1, max_length = 1,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Configures the DM-RS for mapping type B. The settings apply to the <BWP> on carrier <cc> in segment <no>.

param bwp
No help available

param config_type
DM-RS setting 'dmrs-Type'.

param add_position
DM-RS setting 'dmrs-AdditionalPosition'.

param max_length
DM-RS setting 'maxLength'.

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')

6.1.1.5.2.14 Cbandwidth

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:CBANdwidth
```

class CbandwidthCls

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

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → ChannelBwidth

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:CBANdwidth
value: enums.ChannelBwidth = driver.configure.nrSubMeas.listPy.segment.cc.
↪cbandwidth.get(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Specifies the channel bandwidth of carrier <cc>, used in segment <no>.

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
channel_bw: Channel bandwidth 5 MHz to 100 MHz (Bxxx = xxx MHz) .

set(*channel_bw: ChannelBwidth*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:CBANdwidth
driver.configure.nrSubMeas.listPy.segment.cc.cbandwidth.set(channel_bw = enums.
↪ChannelBwidth.B005, sEGMent = repcap.SEGMent.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Specifies the channel bandwidth of carrier <cc>, used in segment <no>.

param channel_bw
Channel bandwidth 5 MHz to 100 MHz (Bxxx = xxx 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')

6.1.1.5.2.15 Frequency

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:FREQuency
```

class FrequencyCls

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

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:FREQuency
value: float = driver.configure.nrSubMeas.listPy.segment.cc.frequency.
↪get(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Selects the center frequency of carrier <cc>, used in segment <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'. For the supported frequency range, see 'Frequency ranges'.

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
frequency: No help available

set(*frequency: float*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:FREQuency
driver.configure.nrSubMeas.listPy.segment.cc.frequency.set(frequency = 1.0,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Selects the center frequency of carrier <cc>, used in segment <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'. For the supported frequency range, see 'Frequency ranges'.

param frequency
No help available

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')

6.1.1.5.2.16 Nallocations

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:NALLocations
```

class NallocationsCls

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

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:NALLocations
value: int = driver.configure.nrSubMeas.listPy.segment.cc.nallocations.
↪get(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Configures the number of allocations on carrier <cc> in segment <no>.

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
number: For the measured carrier, only 1 is allowed.

set(*number: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:NALLocations
driver.configure.nrSubMeas.listPy.segment.cc.nallocations.set(number = 1,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Configures the number of allocations on carrier <cc> in segment <no>.

param number
For the measured carrier, only 1 is allowed.

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')

6.1.1.5.2.17 PlcId

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:PLCid
```

class PlcIdCls

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

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:PLCid
value: int = driver.configure.nrSubMeas.listPy.segment.cc.plcId.get(sEGMent =␣
↪repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.Default)
```

Specifies the physical cell ID of carrier <cc> in segment <no>. See also method RsCMPX_NrFr1Meas.Configure.NrSubMeas. ListPy.PlcId.mode.

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
physical_cell_id: No help available

set(*physical_cell_id: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:PLCid
driver.configure.nrSubMeas.listPy.segment.cc.plcId.set(physical_cell_id = 1,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Specifies the physical cell ID of carrier <cc> in segment <no>. See also method RsCMPX_NrFr1Meas.Configure.NrSubMeas. ListPy.PlcId.mode.

param physical_cell_id
No help available

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')

6.1.1.5.2.18 TaPosition

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:TAPosition
```

class TaPositionCls

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

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:TAPosition
value: int = driver.configure.nrSubMeas.listPy.segment.cc.taPosition.
↪get(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Specifies the 'dmrs-TypeA-Position' for carrier <cc> in segment <no>.

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
position: Number of the first DM-RS symbol for mapping type A.

set(*position: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:TAPosition
driver.configure.nrSubMeas.listPy.segment.cc.taPosition.set(position = 1,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Specifies the 'dmrs-TypeA-Position' for carrier <cc> in segment <no>.

param position
Number of the first DM-RS symbol for mapping type A.

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')

6.1.1.5.2.19 TxBwidth

class TxBwidthCls

TxBwidth 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.nrSubMeas.listPy.segment.cc.txBwidth.clone()
```

Subgroups

6.1.1.5.2.20 Offset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>]:TXBWidth:OFFSet
```

class OffsetCls

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

get(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:TXBWidth:OFFSet
value: int = driver.configure.nrSubMeas.listPy.segment.cc.txBwidth.offset.
↪get(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Specifies the offset to carrier (TxBW offset) for carrier <cc>, used in segment <no>.

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
offset: Number of RBs

set(*offset: int*, *sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>[:CC<cc>
↪]:TXBWidth:OFFSet
driver.configure.nrSubMeas.listPy.segment.cc.txBwidth.offset.set(offset = 1,␣
↪sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.CarrierComponent.
↪Default)
```

Specifies the offset to carrier (TxBW offset) for carrier <cc>, used in segment <no>.

param offset
Number of RBs

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')

6.1.1.5.2.21 Ccall

class CcallCls

Ccall 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.nrSubMeas.listPy.segment.ccall.clone()
```

Subgroups

6.1.1.5.2.22 TxBwidth

class TxBwidthCls

TxBwidth 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.nrSubMeas.listPy.segment.ccall.txBwidth.clone()
```

Subgroups

6.1.1.5.2.23 ScSpacing

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:CCALl:TXBWidth:SCSPacing
```

class ScSpacingCls

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

get(*sEGMent=SEGMent.Default*) → SubCarrSpacing

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>
↪:CCALl:TXBWidth:SCSPacing
value: enums.SubCarrSpacing = driver.configure.nrSubMeas.listPy.segment.ccall.
↪txBwidth.scSpacing.get(sEGMent = repcap.SEGMent.Default)
```

Selects the subcarrier spacing used in segment <no>, for all carriers.

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

return
used_scs: 15 kHz, 30 kHz, 60 kHz In the current software version, you must configure the same value for all segments.

`set`(*used_scs: SubCarrSpacing*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>
↪:CCALl:TXBWidth:SCSPacing
driver.configure.nrSubMeas.listPy.segment.ccall.txBwidth.scSpacing.set(used_scs␣
↪= enums.SubCarrSpacing.S15K, sEGMent = repcap.SEGMent.Default)
```

Selects the subcarrier spacing used in segment <no>, for all carriers.

param used_scs
15 kHz, 30 kHz, 60 kHz In the current software version, you must configure the same value for all segments.

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

6.1.1.5.2.24 Setup

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:SETup
```

`class SetupCls`

Setup commands group definition. 2 total commands, 1 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.DuplexModeB: Duplex mode used in the segment
- Band: enums.Band: TDD UL: OB34 | OB38 | … | OB41 | OB46 | OB47 | OB48 | OB50 | OB51 | OB53 | OB77 | … | OB84 | OB86 | OB89 | OB90 | OB95 | … | OB99 | OB101 | OB104 Operating band used in the segment
- **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.NrSubMeas.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:NRSub: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:NRSub:MEASi:LIST:NBANdwidth.

- Evaluat_Offset: int: Number of subframes at the beginning of the segment that are not evaluated
- Network_Sig_Val: enums.NetworkSigVal: Optional setting parameter. Network signaled value to be used for the segment

get(*sEGMent=SEGMent.Default*) → SetupStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:SETup
value: SetupStruct = driver.configure.nrSubMeas.listPy.segment.setup.
↪get(sEGMent = repcap.SEGMent.Default)
```

Defines the length and analyzer settings of segment <no>. For carrier-specific settings, there are additional commands. This command and the other segment configuration commands must be sent for all segments to be measured (method RsCMPX_NrFr1Meas.Configure.NrSubMeas.MultiEval.ListPy.Lrange.set) .

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:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:SETup
structure = driver.configure.nrSubMeas.listPy.segment.setup.SetupStruct()
structure.Segment_Length: int = 1
structure.Level: float = 1.0
structure.Duplex_Mode: enums.DuplexModeB = enums.DuplexModeB.FDD
structure.Band: enums.Band = enums.Band.OB1
structure.Retrigger_Flag: enums.RetriggerFlag = enums.RetriggerFlag.
↪IFPNarrowband
structure.Evaluat_Offset: int = 1
structure.Network_Sig_Val: enums.NetworkSigVal = enums.NetworkSigVal.NS01
driver.configure.nrSubMeas.listPy.segment.setup.set(structure, sEGMent = repcap.
↪SEGMent.Default)
```

Defines the length and analyzer settings of segment <no>. For carrier-specific settings, there are additional commands. This command and the other segment configuration commands must be sent for all segments to be measured (method RsCMPX_NrFr1Meas.Configure.NrSubMeas.MultiEval.ListPy.Lrange.set) .

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')

Cloning the Group

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

Subgroups

6.1.1.5.2.25 Additional

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:SETup:ADDitional
```

class AdditionalCls

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

get(*sEGMent=SEGMent.Default*) → NbTrigger

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:SETup:ADDitional
value: enums.NbTrigger = driver.configure.nrSubMeas.listPy.segment.setup.
↪additional.get(sEGMent = repcap.SEGMent.Default)
```

No command help available

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

return
narrow_band_trigger: No help available

set(*narrow_band_trigger: NbTrigger*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:SETup:ADDitional
driver.configure.nrSubMeas.listPy.segment.setup.additional.set(narrow_band_
↪trigger = enums.NbTrigger.M010, sEGMent = repcap.SEGMent.Default)
```

No command help available

param narrow_band_trigger
No help available

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

6.1.1.6 MultiEval

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:PFORmat
CONFigure:NRSub:MEASurement<Instance>:MEValuation:TOUT
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMODe
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCSPacing
CONFigure:NRSub:MEASurement<Instance>:MEValuation:CBANdwidth
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DFTPrecoding
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MSCHeme
CONFigure:NRSub:MEASurement<Instance>:MEValuation:CPRefix
CONFigure:NRSub:MEASurement<Instance>:MEValuation:NSValue
CONFigure:NRSub:MEASurement<Instance>:MEValuation:PLCid
CONFigure:NRSub:MEASurement<Instance>:MEValuation:CTYPe
CONFigure:NRSub:MEASurement<Instance>:MEValuation:PUSChconfig
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MAPType
CONFigure:NRSub:MEASurement<Instance>:MEValuation:REPetition
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCONdition
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MMODe
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MOEXception
CONFigure:NRSub:MEASurement<Instance>:MEValuation:NVFilter
CONFigure:NRSub:MEASurement<Instance>:MEValuation:CTVFilter
CONFigure:NRSub:MEASurement<Instance>:MEValuation:NSUBframes
CONFigure:NRSub:MEASurement<Instance>:MEValuation:FSTRucture
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DSSPusch
CONFigure:NRSub:MEASurement<Instance>:MEValuation:GHOPping
```

class MultiEvalCls

MultiEval commands group definition. 150 total commands, 16 Subgroups, 23 group commands

class PuschConfigStruct

Structure for setting input parameters. Fields:

- Mod_Scheme: enums.ModulationScheme: No parameter help available
- Mapping_Type: enums.MappingType: No parameter help available
- Nrb_Auto: bool: No parameter help available
- No_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- No_Symbols: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Config_Type: enums.ConfigType: No parameter help available
- Max_Length: enums.MaxLength: No parameter help available
- Add_Position: int: No parameter help available
- Lzero: int: No parameter help available

get_cbandwidth() → ChannelBwidth

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:CBANdwidth
value: enums.ChannelBwidth = driver.configure.nrSubMeas.multiEval.get_
↪cbandwidth()
```

No command help available

return
channel_bw: No help available

get_cprefix() → CyclicPrefix

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

No command help available

return
cyclic_prefix: No help available

get_ctv_filter() → ChannelTypeB

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:CTVFilter
value: enums.ChannelTypeB = driver.configure.nrSubMeas.multiEval.get_ctv_
↪filter()
```

No command help available

return
channel_type: No help available

get_ctype() → ChannelTypeA

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:CTYPe
value: enums.ChannelTypeA = driver.configure.nrSubMeas.multiEval.get_ctype()
```

No command help available

return
channel_type: No help available

get_dft_precoding() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DFTPrecoding
value: bool = driver.configure.nrSubMeas.multiEval.get_dft_precoding()
```

No command help available

return
on_off: No help available

get_dmode() → DuplexModeB

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMODe
value: enums.DuplexModeB = driver.configure.nrSubMeas.multiEval.get_dmode()
```

Selects the duplex mode of the signal: FDD or TDD.

return
mode: No help available

get_dss_pusch() → int

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

No command help available

return
delta_seq_sh_pusch: No help available

get_fstructure() → ConfigType

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:FSTRucture
value: enums.ConfigType = driver.configure.nrSubMeas.multiEval.get_fstructure()
```

No command help available

return
frame_structure: No help available

get_ghopping() → bool

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

No command help available

return
value: No help available

get_map_type() → MappingType

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MAPType
value: enums.MappingType = driver.configure.nrSubMeas.multiEval.get_map_type()
```

No command help available

return
mapping_type: No help available

get_mmode() → MeasurementMode

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

Selects the measurement mode.

return
measurement_mode: NORMal: normal mode MELMode: multi-evaluation list mode For a setting command, only NORMal is allowed (disables the list mode) . A query can also return MELM.

get_mo_exception() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MOEXception
value: bool = driver.configure.nrSubMeas.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_mscheme() → ModulationScheme

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MSCHeme
value: enums.ModulationScheme = driver.configure.nrSubMeas.multiEval.get_
↪mscheme()
```

No command help available

return
mod_scheme: No help available

get_ns_value() → NetworkSigVal

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:NSValue
value: enums.NetworkSigVal = driver.configure.nrSubMeas.multiEval.get_ns_value()
```

Selects the 'network signaled value'.

return
value: Value NS_01 to NS_100, NS_03U, NS_05U, NS_43U

get_nsub_frames() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:NSUBframes
value: int = driver.configure.nrSubMeas.multiEval.get_nsub_frames()
```

Specifies the number of subframes to be evaluated.

return
no_subframe: No help available

get_nvfilter() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:NVFilter
value: int or bool = driver.configure.nrSubMeas.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.

return
nrb_view_filter: (integer or boolean) Number of allocated resource blocks The allowed values depend on the SC spacing and on the channel bandwidth, see 'Resource elements, grids and blocks'.

get_pformat() → PucchFormat

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

No command help available

return
pucch_format: No help available

get_plc_id() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PLCid
value: int = driver.configure.nrSubMeas.multiEval.get_plc_id()
```

No command help available

return
phs_layer_cell_id: No help available

get_pusch_config() → PuschConfigStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PUSChconfig
value: PuschConfigStruct = driver.configure.nrSubMeas.multiEval.get_pusch_
↪config()
```

No command help available

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

get_repetition() → Repeat

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:REPetition
value: enums.Repeat = driver.configure.nrSubMeas.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_sc_spacing() → SubCarrSpacing

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCSPacing
value: enums.SubCarrSpacing = driver.configure.nrSubMeas.multiEval.get_sc_
↪spacing()
```

No command help available

return
sub_carr_spacing: No help available

get_scondition() → StopCondition

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCONdition
value: enums.StopCondition = driver.configure.nrSubMeas.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_timeout() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:TOUT
value: float = driver.configure.nrSubMeas.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

set_cbandwidth(*channel_bw: ChannelBwidth*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:CBANdwidth
driver.configure.nrSubMeas.multiEval.set_cbandwidth(channel_bw = enums.
↪ChannelBwidth.B005)
```

No command help available

param channel_bw
No help available

set_cprefix(*cyclic_prefix: CyclicPrefix*) → None

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

No command help available

param cyclic_prefix
No help available

set_ctv_filter(*channel_type: ChannelTypeB*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:CTVFilter
driver.configure.nrSubMeas.multiEval.set_ctv_filter(channel_type = enums.
↪ChannelTypeB.OFF)
```

No command help available

param channel_type
No help available

set_ctype(*channel_type: ChannelTypeA*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:CTYPe
driver.configure.nrSubMeas.multiEval.set_ctype(channel_type = enums.
↪ChannelTypeA.PUCCh)
```

No command help available

param channel_type
No help available

set_dft_precoding(*on_off: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DFTPrecoding
driver.configure.nrSubMeas.multiEval.set_dft_precoding(on_off = False)
```

No command help available

param on_off
No help available

set_dmode(*mode: DuplexModeB*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMODe
driver.configure.nrSubMeas.multiEval.set_dmode(mode = enums.DuplexModeB.FDD)
```

Selects the duplex mode of the signal: FDD or TDD.

param mode
No help available

set_dss_pusch(*delta_seq_sh_pusch: int*) → None

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

No command help available

param delta_seq_sh_pusch
No help available

set_ghopping(*value: bool*) → None

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

No command help available

param value
No help available

set_map_type(*mapping_type: MappingType*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MAPType
driver.configure.nrSubMeas.multiEval.set_map_type(mapping_type = enums.
↪MappingType.A)
```

No command help available

param mapping_type
No help available

set_mmode(*measurement_mode: MeasurementMode*) → None

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

Selects the measurement mode.

param measurement_mode
NORMal: normal mode MELMode: multi-evaluation list mode For a setting command, only NORMal is allowed (disables the list mode) . A query can also return MELM.

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MOEXception
driver.configure.nrSubMeas.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_mscheme(*mod_scheme: ModulationScheme*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MSCHeme
driver.configure.nrSubMeas.multiEval.set_mscheme(mod_scheme = enums.
↪ModulationScheme.AUTO)
```

No command help available

param mod_scheme
No help available

set_ns_value(*value: NetworkSigVal*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:NSValue
driver.configure.nrSubMeas.multiEval.set_ns_value(value = enums.NetworkSigVal.
↪NS01)
```

Selects the 'network signaled value'.

param value
Value NS_01 to NS_100, NS_03U, NS_05U, NS_43U

set_nsub_frames(*no_subframe: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:NSUBframes
driver.configure.nrSubMeas.multiEval.set_nsub_frames(no_subframe = 1)
```

Specifies the number of subframes to be evaluated.

param no_subframe
No help available

set_nvfilter(*nrb_view_filter: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:NVFilter
driver.configure.nrSubMeas.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.

param nrb_view_filter
(integer or boolean) Number of allocated resource blocks The allowed values depend

on the SC spacing and on the channel bandwidth, see 'Resource elements, grids and blocks'.

set_pformat(*pucch_format: PucchFormat*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PFORmat
driver.configure.nrSubMeas.multiEval.set_pformat(pucch_format = enums.
↪PucchFormat.F0)
```

No command help available

param pucch_format
No help available

set_plc_id(*phs_layer_cell_id: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PLCid
driver.configure.nrSubMeas.multiEval.set_plc_id(phs_layer_cell_id = 1)
```

No command help available

param phs_layer_cell_id
No help available

set_pusch_config(*value: PuschConfigStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PUSChconfig
structure = driver.configure.nrSubMeas.multiEval.PuschConfigStruct()
structure.Mod_Scheme: enums.ModulationScheme = enums.ModulationScheme.AUTO
structure.Mapping_Type: enums.MappingType = enums.MappingType.A
structure.Nrb_Auto: bool = False
structure.No_Rb: int = 1
structure.Start_Rb: int = 1
structure.No_Symbols: int = 1
structure.Start_Symbol: int = 1
structure.Config_Type: enums.ConfigType = enums.ConfigType.T1
structure.Max_Length: enums.MaxLength = enums.MaxLength.DOUBle
structure.Add_Position: int = 1
structure.Lzero: int = 1
driver.configure.nrSubMeas.multiEval.set_pusch_config(value = structure)
```

No command help available

param value
see the help for PuschConfigStruct structure arguments.

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:REPetition
driver.configure.nrSubMeas.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_sc_spacing(*sub_carr_spacing: SubCarrSpacing*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCSPacing
driver.configure.nrSubMeas.multiEval.set_sc_spacing(sub_carr_spacing = enums.
↪SubCarrSpacing.S15K)
```

No command help available

param sub_carr_spacing
No help available

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCONdition
driver.configure.nrSubMeas.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_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:TOUT
driver.configure.nrSubMeas.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

Cloning the Group

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

Subgroups

6.1.1.6.1 Allocation

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:ALLocation:NSYMbols
CONFigure:NRSub:MEASurement<Instance>:MEValuation:ALLocation:SSYMbol
```

class AllocationCls

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

get_nsymbols() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ALLocation:NSYMbols
value: int = driver.configure.nrSubMeas.multiEval.allocation.get_nsymbols()
```

No command help available

return
no_symbols: No help available

get_ssymbol() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ALLocation:SSYMbol
value: int = driver.configure.nrSubMeas.multiEval.allocation.get_ssymbol()
```

No command help available

return
start_symbol: No help available

set_nsymbols(*no_symbols: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ALLocation:NSYMbols
driver.configure.nrSubMeas.multiEval.allocation.set_nsymbols(no_symbols = 1)
```

No command help available

param no_symbols
No help available

set_ssymbol(*start_symbol: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ALLocation:SSYMbol
driver.configure.nrSubMeas.multiEval.allocation.set_ssymbol(start_symbol = 1)
```

No command help available

param start_symbol
No help available

6.1.1.6.2 BwConfig

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:BWConfig
```

class BwConfigCls

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

class BwConfigStruct

Response structure. Fields:

- Sub_Carr_Spacing: enums.SubCarrSpacing: No parameter help available
- Channel_Bw: enums.ChannelBwidth: No parameter help available

get() → BwConfigStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:BWConfig
value: BwConfigStruct = driver.configure.nrSubMeas.multiEval.bwConfig.get()
```

No command help available

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

set(*sub_carr_spacing: SubCarrSpacing*, *channel_bw: ChannelBwidth*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:BWConfig
driver.configure.nrSubMeas.multiEval.bwConfig.set(sub_carr_spacing = enums.
↪SubCarrSpacing.S15K, channel_bw = enums.ChannelBwidth.B005)
```

No command help available

param sub_carr_spacing
No help available

param channel_bw
No help available

6.1.1.6.3 Dmrs

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:CONFigtype
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:MAXLength
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:APOSition
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:LZERo
```

class DmrsCls

Dmrs commands group definition. 5 total commands, 1 Subgroups, 4 group commands

get_aposition() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:APOSition
value: int = driver.configure.nrSubMeas.multiEval.dmrs.get_aposition()
```

No command help available

return
add_position: No help available

get_config_type() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:CONFigtype
value: int = driver.configure.nrSubMeas.multiEval.dmrs.get_config_type()
```

No command help available

return
config_type: No help available

get_lzero() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:LZERo
value: int = driver.configure.nrSubMeas.multiEval.dmrs.get_lzero()
```

No command help available

return
lzero: No help available

get_max_length() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:MAXLength
value: int = driver.configure.nrSubMeas.multiEval.dmrs.get_max_length()
```

No command help available

return
max_length: No help available

set_aposition(*add_position: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:APOSition
driver.configure.nrSubMeas.multiEval.dmrs.set_aposition(add_position = 1)
```

No command help available

param add_position
No help available

set_lzero(*lzero: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:LZERo
driver.configure.nrSubMeas.multiEval.dmrs.set_lzero(lzero = 1)
```

No command help available

param lzero
No help available

set_max_length(*max_length: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:MAXLength
driver.configure.nrSubMeas.multiEval.dmrs.set_max_length(max_length = 1)
```

No command help available

param max_length
No help available

Cloning the Group

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

Subgroups

6.1.1.6.3.1 Sgeneration

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:SGENeration
```

class SgenerationCls

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

class SgenerationStruct

Response structure. Fields:

- Generator: enums.Generator: No parameter help available
- Dmrs_Id: int: No parameter help available
- Scid: int: No parameter help available

get() → SgenerationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:SGENeration
value: SgenerationStruct = driver.configure.nrSubMeas.multiEval.dmrs.
↪sgeneration.get()
```

No command help available

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

set(*generator: Generator*, *dmrs_id: int*, *scid: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:DMRS:SGENeration
driver.configure.nrSubMeas.multiEval.dmrs.sgeneration.set(generator = enums.
↪Generator.DID, dmrs_id = 1, scid = 1)
```

No command help available

param generator
No help available

param dmrs_id
No help available

param scid
No help available

6.1.1.6.4 Endc

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:ENDC
```

class EndcCls

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

get_value() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ENDC
value: bool = driver.configure.nrSubMeas.multiEval.endc.get_value()
```

Enables or disables the EN-DC mode of the measurement.

return
on_off: No help available

set_value(*on_off: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ENDC
driver.configure.nrSubMeas.multiEval.endc.set_value(on_off = False)
```

Enables or disables the EN-DC mode of the measurement.

param on_off
No help available

Cloning the Group

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

Subgroups

6.1.1.6.4.1 Eutra

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:ENDC:EUTRa
```

class EutraCls

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

class EutraStruct

Response structure. Fields:

- Channel_Bw: enums.ChannelBwidthB: Channel bandwidth in MHz (5 MHz to 20 MHz) .
- Carrier_Position: enums.CarrierPosition: Position of LTE carrier left (LONR) or right (RONR) of NR carrier.

get() → EutraStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ENDC:EUTRa
value: EutraStruct = driver.configure.nrSubMeas.multiEval.endc.eutra.get()
```

Configures LTE settings for EN-DC.

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

set(*channel_bw: ChannelBwidthB*, *carrier_position: CarrierPosition*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:ENDC:EUTRa
driver.configure.nrSubMeas.multiEval.endc.eutra.set(channel_bw = enums.
↪ChannelBwidthB.B005, carrier_position = enums.CarrierPosition.LONR)
```

Configures LTE settings for EN-DC.

param channel_bw
Channel bandwidth in MHz (5 MHz to 20 MHz) .

param carrier_position
Position of LTE carrier left (LONR) or right (RONR) of NR carrier.

6.1.1.6.5 Limit

class LimitCls

Limit commands group definition. 55 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.6.5.1 Aclr

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:ENDC
```

class AclrCls

Aclr commands group definition. 5 total commands, 4 Subgroups, 1 group commands

get_endc() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:ENDC
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.aclr.get_
↪endc()
```

Defines a relative limit for the ACLR measured in an adjacent channel in EN-DC mode.

return
relative_level: (float or boolean) Relative lower ACLR limit without test tolerance

set_endc(*relative_level: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:ENDC
driver.configure.nrSubMeas.multiEval.limit.aclr.set_endc(relative_level = 1.0)
```

Defines a relative limit for the ACLR measured in an adjacent channel in EN-DC mode.

param relative_level
(float or boolean) Relative lower ACLR limit without test tolerance

Cloning the Group

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

Subgroups

6.1.1.6.5.2 Caggregation

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:CAGGregation
```

class CaggregationCls

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

class CaggregationStruct

Response structure. Fields:

- Relative_Level: float or bool: Relative lower ACLR limit without test tolerance

- Absolute_Level: float or bool: No parameter help available

get() → CaggregationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:CAGGregation
value: CaggregationStruct = driver.configure.nrSubMeas.multiEval.limit.aclr.
↪caggregation.get()
```

Defines relative and absolute limits for the ACLR measured in an adjacent aggregated channel bandwidth, for NR SA with carrier aggregation.

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:CAGGregation
driver.configure.nrSubMeas.multiEval.limit.aclr.caggregation.set(relative_level␣
↪= 1.0, absolute_level = 1.0)
```

Defines relative and absolute limits for the ACLR measured in an adjacent aggregated channel bandwidth, for NR SA with carrier aggregation.

param relative_level
(float or boolean) Relative lower ACLR limit without test tolerance

param absolute_level
(float or boolean) No help available

6.1.1.6.5.3 Nr

class NrCls

Nr 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.nrSubMeas.multiEval.limit.aclr.nr.clone()
```

Subgroups

6.1.1.6.5.4 Cbandwidth<ChannelBw>

RepCap Settings

```
# Range: Bw5 .. Bw100
rc = driver.configure.nrSubMeas.multiEval.limit.aclr.nr.cbandwidth.repcap_channelBw_get()
driver.configure.nrSubMeas.multiEval.limit.aclr.nr.cbandwidth.repcap_channelBw_
↪set(repcap.ChannelBw.Bw5)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:NR:CBANdwidth<bw>
```

class CbandwidthCls

Cbandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw5

class CbandwidthStruct

Response structure. Fields:

- Relative_Level: float or bool: Relative lower ACLR limit without test tolerance
- Absolute_Level: float or bool: No parameter help available

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:NR:CBANdwidth<bw>
value: CbandwidthStruct = driver.configure.nrSubMeas.multiEval.limit.aclr.nr.
↪cbandwidth.get(channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in an adjacent NR channel (for NR SA without CA) . The settings are defined separately for each channel bandwidth.

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:ACLR:NR:CBANdwidth<bw>
driver.configure.nrSubMeas.multiEval.limit.aclr.nr.cbandwidth.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 NR channel (for NR SA without CA) . The settings are defined separately for each channel bandwidth.

param relative_level
(float or boolean) Relative lower ACLR limit without test tolerance

param absolute_level
(float or boolean) No help available

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

Cloning the Group

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

6.1.1.6.5.5 Ttolerance

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:TTOLerance
```

class TtoleranceCls

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

class TtoleranceStruct

Response structure. Fields:

- Test_Tol_Sub_4_Ghz: float: Test tolerance for center frequencies 4 GHz
- Test_Tol_Sub_6_Gh_Z: float: Test tolerance for center frequencies 4 GHz

get() → TtoleranceStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:TTOLerance
value: TtoleranceStruct = driver.configure.nrSubMeas.multiEval.limit.aclr.
↪ttolerance.get()
```

Defines the test tolerance for relative ACLR limits, depending on the center frequency.

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

set(*test_tol_sub_4_ghz: float*, *test_tol_sub_6_gh_z: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:TTOLerance
driver.configure.nrSubMeas.multiEval.limit.aclr.ttolerance.set(test_tol_sub_4_
↪ghz = 1.0, test_tol_sub_6_gh_z = 1.0)
```

Defines the test tolerance for relative ACLR limits, depending on the center frequency.

param test_tol_sub_4_ghz
Test tolerance for center frequencies 4 GHz

param test_tol_sub_6_gh_z
Test tolerance for center frequencies 4 GHz

6.1.1.6.5.6 Utra<UtraChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.nrSubMeas.multiEval.limit.aclr.utra.repcap_utraChannel_get()
driver.configure.nrSubMeas.multiEval.limit.aclr.utra.repcap_utraChannel_set(repcap.
↪UtraChannel.Nr1)
```

class UtraCls

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

Cloning the Group

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

Subgroups

6.1.1.6.5.7 Cbandwidth<ChannelBw>

RepCap Settings

```
# Range: Bw5 .. Bw100
rc = driver.configure.nrSubMeas.multiEval.limit.aclr.utra.cbandwidth.repcap_channelBw_
↪get()
driver.configure.nrSubMeas.multiEval.limit.aclr.utra.cbandwidth.repcap_channelBw_
↪set(repcap.ChannelBw.Bw5)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<no>:CBANdwidth<bw>
```

class CbandwidthCls

Cbandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw5

class CbandwidthStruct

Response structure. Fields:

- Relative_Level: float or bool: Relative lower ACLR limit without test tolerance
- Absolute_Level: float or bool: No parameter help available

get(*utraChannel=UtraChannel.Default*, *channelBw=ChannelBw.Default*) → CbandwidthStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<no>
↪:CBANdwidth<bw>
value: CbandwidthStruct = driver.configure.nrSubMeas.multiEval.limit.aclr.utra.
↪cbandwidth.get(utraChannel = repcap.UtraChannel.Default, channelBw = repcap.
↪ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on UTRA<no> (for NR SA without CA) . The settings are defined separately for each channel bandwidth.

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

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:ACLR:UTRA<no>
↪:CBANdwidth<bw>
driver.configure.nrSubMeas.multiEval.limit.aclr.utra.cbandwidth.set(relative_
↪level = 1.0, absolute_level = 1.0, utraChannel = repcap.UtraChannel.Default,␣
↪channelBw = repcap.ChannelBw.Default)
```

Defines relative and absolute limits for the ACLR measured in the first or second adjacent UTRA channel, depending on UTRA<no> (for NR SA without CA) . The settings are defined separately for each channel bandwidth.

param relative_level
(float or boolean) Relative lower ACLR limit without test tolerance

param absolute_level
(float or boolean) No help available

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

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

Cloning the Group

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

6.1.1.6.5.8 BpwShaping

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping:FERRor
```

class BpwShapingCls

BpwShaping commands group definition. 8 total commands, 6 Subgroups, 1 group commands

get_freq_error() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:FERRor
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.
↪get_freq_error()
```

Defines an upper limit for the carrier frequency error (/2-BPSK modulation with shaping) .

return
frequency_error: (float or boolean) No help available

set_freq_error(*frequency_error: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:FERRor
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.set_freq_error(frequency_
↪error = 1.0)
```

Defines an upper limit for the carrier frequency error (/2-BPSK modulation with shaping) .

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.nrSubMeas.multiEval.limit.bpwShaping.clone()
```

Subgroups

6.1.1.6.5.9 EsFlatness

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping:ESFLatness
```

class EsFlatnessCls

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

class EsFlatnessStruct

Response structure. 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)

get() → EsFlatnessStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:ESFLatness
value: EsFlatnessStruct = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.
↪esFlatness.get()
```

Defines limits for the equalizer spectrum flatness (/2-BPSK modulation with shaping) .

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

set(*enable: bool*, *range_1: float*, *range_2: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:ESFLatness
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.esFlatness.set(enable =␣
↪False, range_1 = 1.0, range_2 = 1.0)
```

Defines limits for the equalizer spectrum flatness (/2-BPSK modulation with shaping) .

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

param range_1
Upper limit for max(range 1) - min(range 1)

param range_2
Upper limit for max(range 2) - min(range 2)

6.1.1.6.5.10 EvMagnitude

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:EVMagnitude
value: EvMagnitudeStruct = driver.configure.nrSubMeas.multiEval.limit.
↪bpwShaping.evMagnitude.get()
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for /2-BPSK with shaping.

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:EVMagnitude
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.evMagnitude.set(rms = 1.0,
↪ peak = 1.0)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for /2-BPSK with shaping.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.6.5.11 Ibe

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping:IBE
```

class IbeCls

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

class ValueStruct

Structure for setting input parameters. 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_Lesser: float: I/Q image for low TX power range
- Iq_Image_Greater: float: I/Q image for high TX power range

get_value() → ValueStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping:IBE
value: ValueStruct = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.ibe.
↪get_value()
```

Defines parameters used for calculation of an upper limit for the in-band emission (/2-BPSK modulation with shaping) , see 'In-band emissions limits'.

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

set_value(*value: ValueStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping:IBE
structure = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.ibe.
↪ValueStruct()
structure.Enable: bool = False
structure.Minimum: float = 1.0
structure.Evm: float = 1.0
structure.Rb_Power: float = 1.0
structure.Iq_Image_Lesser: float = 1.0
structure.Iq_Image_Greater: float = 1.0
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.ibe.set_value(value =␣
↪structure)
```

Defines parameters used for calculation of an upper limit for the in-band emission (/2-BPSK modulation with shaping) , see 'In-band emissions limits'.

param value
see the help for ValueStruct structure arguments.

Cloning the Group

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

Subgroups

6.1.1.6.5.12 IqOffset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping:IBE:IQOFfset
```

class IqOffsetCls

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

class IqOffsetStruct

Response structure. Fields:

- Offset_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get() → IqOffsetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:IBE:IQOFfset
value: IqOffsetStruct = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.
↪ibe.iqOffset.get()
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission, for /2-BPSK modulation with shaping. Four different values can be set for four TX power ranges.

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

set(*offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:IBE:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.ibe.iqOffset.set(offset_0␣
↪= 1.0, 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, for /2-BPSK modulation with shaping. Four different values can be set for four TX power ranges.

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

6.1.1.6.5.13 IqOffset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping: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_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get() → IqOffsetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:IQOFfset
value: IqOffsetStruct = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.
↪iqOffset.get()
```

Defines upper limits for the I/Q origin offset (/2-BPSK modulation with shaping) . Four different I/Q origin offset limits can be set for four TX power ranges.

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

set(*enable: bool*, *offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.iqOffset.set(enable =␣
↪False, offset_0 = 1.0, offset_1 = 1.0, offset_2 = 1.0, offset_3 = 1.0)
```

Defines upper limits for the I/Q origin offset (/2-BPSK modulation with shaping) . Four different I/Q origin offset limits can be set for four TX power ranges.

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

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

6.1.1.6.5.14 Merror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:MERRor
value: MerrorStruct = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.
↪merror.get()
```

Defines upper limits for the RMS and peak values of the magnitude error for /2-BPSK with shaping.

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:MERRor
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.merror.set(rms = 1.0,␣
↪peak = 1.0)
```

Defines upper limits for the RMS and peak values of the magnitude error for /2-BPSK with shaping.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.6.5.15 Perror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:BPWShaping: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:PERRor
value: PerrorStruct = driver.configure.nrSubMeas.multiEval.limit.bpwShaping.
↪perror.get()
```

Defines symmetric limits for the RMS and peak values of the phase error for /2-BPSK with shaping. 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:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:BPWShaping:PERRor
driver.configure.nrSubMeas.multiEval.limit.bpwShaping.perror.set(rms = 1.0,␣
↪peak = 1.0)
```

Defines symmetric limits for the RMS and peak values of the phase error for /2-BPSK with shaping. 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

6.1.1.6.5.16 Pdynamics

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:ENABle
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:OFFPower
```

class PdynamicsCls

Pdynamics commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_enable() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:ENABle
value: bool = driver.configure.nrSubMeas.multiEval.limit.pdynamics.get_enable()
```

Enables or disables the limit check for the power dynamics measurement.

return
enable: No help available

get_off_power() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:OFFPower
```

(continues on next page)

(continued from previous page)

```
value: float = driver.configure.nrSubMeas.multiEval.limit.pdynamics.get_off_
↪power()
```

Defines an upper limit for the OFF power determined with the power dynamics measurement.

return
power: No help available

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:ENABle
driver.configure.nrSubMeas.multiEval.limit.pdynamics.set_enable(enable = False)
```

Enables or disables the limit check for the power dynamics measurement.

param enable
No help available

set_off_power(*power: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:OFFPower
driver.configure.nrSubMeas.multiEval.limit.pdynamics.set_off_power(power = 1.0)
```

Defines an upper limit for the OFF power determined with the power dynamics measurement.

param power
No help available

Cloning the Group

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

Subgroups

6.1.1.6.5.17 Ttolerance

class TtoleranceCls

Ttolerance 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.nrSubMeas.multiEval.limit.pdynamics.ttolerance.clone()
```

Subgroups

6.1.1.6.5.18 Cbgt

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:TTOLerance:CBGT
```

class CbgtCls

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

class CbgtStruct

Response structure. Fields:

- Tt_Power_Less_3_G: float: Tolerance for carrier center frequencies up to 3 GHz
- Tt_Power_Great_3_G: float: Tolerance for carrier center frequencies 3 GHz

get() → CbgtStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:TTOLerance:CBGT
value: CbgtStruct = driver.configure.nrSubMeas.multiEval.limit.pdynamics.
↪ttolerance.cbgt.get()
```

Defines test tolerances for power dynamics limits, for channel BW > 40 MHz.

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

set(*tt_power_less_3_g: float*, *tt_power_great_3_g: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:TTOLerance:CBGT
driver.configure.nrSubMeas.multiEval.limit.pdynamics.ttolerance.cbgt.set(tt_
↪power_less_3_g = 1.0, tt_power_great_3_g = 1.0)
```

Defines test tolerances for power dynamics limits, for channel BW > 40 MHz.

param tt_power_less_3_g
Tolerance for carrier center frequencies up to 3 GHz

param tt_power_great_3_g
Tolerance for carrier center frequencies 3 GHz

6.1.1.6.5.19 Cblt

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PDYNamics:TTOLerance:CBLT
```

class CbltCls

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

class CbltStruct

Response structure. Fields:

- Tt_Power_Less_3_G: float: Tolerance for carrier center frequencies up to 3 GHz
- Tt_Power_Great_3_G: float: Tolerance for carrier center frequencies 3 GHz

get() → CbltStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:TTOLerance:CBLT
value: CbltStruct = driver.configure.nrSubMeas.multiEval.limit.pdynamics.
↪ttolerance.cblt.get()
```

Defines test tolerances for power dynamics limits, for channel BW up to 40 MHz.

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

set(*tt_power_less_3_g: float*, *tt_power_great_3_g: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PDYNamics:TTOLerance:CBLT
driver.configure.nrSubMeas.multiEval.limit.pdynamics.ttolerance.cblt.set(tt_
↪power_less_3_g = 1.0, tt_power_great_3_g = 1.0)
```

Defines test tolerances for power dynamics limits, for channel BW up to 40 MHz.

param tt_power_less_3_g
Tolerance for carrier center frequencies up to 3 GHz

param tt_power_great_3_g
Tolerance for carrier center frequencies 3 GHz

6.1.1.6.5.20 Phbpsk

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:FERRor
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:ESFLatness
```

class PhbpskCls

Phbpsk commands group definition. 8 total commands, 5 Subgroups, 2 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: Band edge distance of border between range 1 and range 2

get_es_flatness() → EsFlatnessStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PHBPsk:ESFLatness
value: EsFlatnessStruct = driver.configure.nrSubMeas.multiEval.limit.phbpsk.get_
↪es_flatness()
```

Defines limits for the equalizer spectrum flatness (/2-BPSK modulation) .

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

get_freq_error() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:FERRor
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.phbpsk.get_
↪freq_error()
```

Defines an upper limit for the carrier frequency error (/2-BPSK modulation) .

return
frequency_error: (float or boolean) No help available

set_es_flatness(*value: EsFlatnessStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PHBPsk:ESFLatness
structure = driver.configure.nrSubMeas.multiEval.limit.phbpsk.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.nrSubMeas.multiEval.limit.phbpsk.set_es_flatness(value =␣
↪structure)
```

Defines limits for the equalizer spectrum flatness (/2-BPSK modulation) .

param value
see the help for EsFlatnessStruct structure arguments.

set_freq_error(*frequency_error: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:FERRor
driver.configure.nrSubMeas.multiEval.limit.phbpsk.set_freq_error(frequency_
↪error = 1.0)
```

Defines an upper limit for the carrier frequency error (/2-BPSK modulation) .

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.nrSubMeas.multiEval.limit.phbpsk.clone()
```

Subgroups

6.1.1.6.5.21 EvMagnitude

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PHBPsk:EVMagnitude
value: EvMagnitudeStruct = driver.configure.nrSubMeas.multiEval.limit.phbpsk.
↪evMagnitude.get()
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for /2-BPSK.

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PHBPsk:EVMagnitude
driver.configure.nrSubMeas.multiEval.limit.phbpsk.evMagnitude.set(rms = 1.0,␣
↪peak = 1.0)
```

Defines upper limits for the RMS and peak values of the error vector magnitude (EVM) for /2-BPSK.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.6.5.22 Ibe

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:IBE
```

class IbeCls

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

class ValueStruct

Structure for setting input parameters. 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_Lesser: float: I/Q image for low TX power range
- Iq_Image_Greater: float: I/Q image for high TX power range

get_value() → ValueStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:IBE
value: ValueStruct = driver.configure.nrSubMeas.multiEval.limit.phbpsk.ibe.get_
↪value()
```

Defines parameters used for calculation of an upper limit for the in-band emission (/2-BPSK modulation) , see 'In-band emissions limits'.

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

set_value(*value: ValueStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:IBE
structure = driver.configure.nrSubMeas.multiEval.limit.phbpsk.ibe.ValueStruct()
structure.Enable: bool = False
structure.Minimum: float = 1.0
structure.Evm: float = 1.0
structure.Rb_Power: float = 1.0
structure.Iq_Image_Lesser: float = 1.0
structure.Iq_Image_Greater: float = 1.0
driver.configure.nrSubMeas.multiEval.limit.phbpsk.ibe.set_value(value =␣
↪structure)
```

Defines parameters used for calculation of an upper limit for the in-band emission (/2-BPSK modulation) , see 'In-band emissions limits'.

param value
see the help for ValueStruct structure arguments.

Cloning the Group

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

Subgroups

6.1.1.6.5.23 IqOffset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:IBE:IQOFfset
```

class IqOffsetCls

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

class IqOffsetStruct

Response structure. Fields:

- Offset_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get() → IqOffsetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PHBPsk:IBE:IQOFfset
value: IqOffsetStruct = driver.configure.nrSubMeas.multiEval.limit.phbpsk.ibe.
↪iqOffset.get()
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission, for /2-BPSK modulation. Four different values can be set for four TX power ranges.

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

set(*offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:PHBPsk:IBE:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.phbpsk.ibe.iqOffset.set(offset_0 = 1.
↪0, 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, for /2-BPSK modulation. Four different values can be set for four TX power ranges.

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

6.1.1.6.5.24 IqOffset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk: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_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get() → IqOffsetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:IQOFfset
value: IqOffsetStruct = driver.configure.nrSubMeas.multiEval.limit.phbpsk.
↪iqOffset.get()
```

Defines upper limits for the I/Q origin offset (/2-BPSK modulation) . Four different I/Q origin offset limits can be set for four TX power ranges.

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

set(*enable: bool*, *offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.phbpsk.iqOffset.set(enable = False,␣
↪offset_0 = 1.0, offset_1 = 1.0, offset_2 = 1.0, offset_3 = 1.0)
```

Defines upper limits for the I/Q origin offset (/2-BPSK modulation) . Four different I/Q origin offset limits can be set for four TX power ranges.

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

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

6.1.1.6.5.25 Merror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk: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:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:MERRor
value: MerrorStruct = driver.configure.nrSubMeas.multiEval.limit.phbpsk.merror.
↪get()
```

Defines upper limits for the RMS and peak values of the magnitude error for /2-BPSK.

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:MERRor
driver.configure.nrSubMeas.multiEval.limit.phbpsk.merror.set(rms = 1.0, peak =␣
↪1.0)
```

Defines upper limits for the RMS and peak values of the magnitude error for /2-BPSK.

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.6.5.26 Perror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk: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:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:PERRor
value: PerrorStruct = driver.configure.nrSubMeas.multiEval.limit.phbpsk.perror.
↪get()
```

Defines symmetric limits for the RMS and peak values of the phase error for /2-BPSK. 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:NRSub:MEASurement<Instance>:MEValuation:LIMit:PHBPsk:PERRor
driver.configure.nrSubMeas.multiEval.limit.phbpsk.perror.set(rms = 1.0, peak =␣
↪1.0)
```

Defines symmetric limits for the RMS and peak values of the phase error for /2-BPSK. 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

6.1.1.6.5.27 Qam<Qam>

RepCap Settings

```
# Range: Order16 .. Order256
rc = driver.configure.nrSubMeas.multiEval.limit.qam.repcap_qam_get()
driver.configure.nrSubMeas.multiEval.limit.qam.repcap_qam_set(repcap.Qam.Order16)
```

class QamCls

Qam commands group definition. 8 total commands, 7 Subgroups, 0 group commands Repeated Capability: Qam, default value after init: Qam.Order16

Cloning the Group

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

Subgroups

6.1.1.6.5.28 EsFlatness

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>: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: Band edge distance of border between range 1 and range 2

get(*qam=Qam.Default*) → EsFlatnessStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:ESFLatness
value: EsFlatnessStruct = driver.configure.nrSubMeas.multiEval.limit.qam.
↪esFlatness.get(qam = repcap.Qam.Default)
```

Defines limits for the equalizer spectrum flatness (QAM modulations) .

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

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

set(*structure: EsFlatnessStruct*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:ESFLatness
structure = driver.configure.nrSubMeas.multiEval.limit.qam.esFlatness.
↪EsFlatnessStruct()
structure.Enable: bool = False
structure.Range_1: float = 1.0
structure.Range_2: float = 1.0
```

(continues on next page)

(continued from previous page)

```
structure.Max_1_Min_2: float = 1.0
structure.Max_2_Min_1: float = 1.0
structure.Edge_Frequency: float = 1.0
driver.configure.nrSubMeas.multiEval.limit.qam.esFlatness.set(structure, qam =␣
↪repcap.Qam.Default)
```

Defines limits for the equalizer spectrum flatness (QAM modulations) .

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

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

6.1.1.6.5.29 EvMagnitude

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>: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(*qam=Qam.Default*) → EvMagnitudeStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:EVMagnitude
value: EvMagnitudeStruct = driver.configure.nrSubMeas.multiEval.limit.qam.
↪evMagnitude.get(qam = repcap.Qam.Default)
```

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

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:EVMagnitude
driver.configure.nrSubMeas.multiEval.limit.qam.evMagnitude.set(rms = 1.0, peak␣
↪= 1.0, qam = repcap.Qam.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 qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

6.1.1.6.5.30 FreqError

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>:FERRor
```

class FreqErrorCls

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

get(*qam=Qam.Default*) → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:FERRor
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.qam.freqError.
↪get(qam = repcap.Qam.Default)
```

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

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

return
frequency_error: (float or boolean) No help available

set(*frequency_error: float*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:FERRor
driver.configure.nrSubMeas.multiEval.limit.qam.freqError.set(frequency_error =␣
↪1.0, qam = repcap.Qam.Default)
```

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

param frequency_error
(float or boolean) No help available

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

6.1.1.6.5.31 Ibe

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>:IBE
```

class IbeCls

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

class IbeStruct

Structure for setting input parameters. 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_Lesser: float: I/Q image for low TX power range
- Iq_Image_Greater: float: I/Q image for high TX power range

get(*qam=Qam.Default*) → IbeStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>:IBE
value: IbeStruct = driver.configure.nrSubMeas.multiEval.limit.qam.ibe.get(qam =␣
↪repcap.Qam.Default)
```

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

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

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

set(*structure: IbeStruct*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>:IBE
structure = driver.configure.nrSubMeas.multiEval.limit.qam.ibe.IbeStruct()
structure.Enable: bool = False
structure.Minimum: float = 1.0
structure.Evm: float = 1.0
structure.Rb_Power: float = 1.0
structure.Iq_Image_Lesser: float = 1.0
structure.Iq_Image_Greater: float = 1.0
driver.configure.nrSubMeas.multiEval.limit.qam.ibe.set(structure, qam = repcap.
↪Qam.Default)
```

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

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

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

Cloning the Group

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

Subgroups

6.1.1.6.5.32 IqOffset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>:IBE:IQOFfset
```

class IqOffsetCls

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

class IqOffsetStruct

Response structure. Fields:

- Offset_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get(*qam=Qam.Default*) → IqOffsetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:IBE:IQOFfset
value: IqOffsetStruct = driver.configure.nrSubMeas.multiEval.limit.qam.ibe.
↪iqOffset.get(qam = repcap.Qam.Default)
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission, for QAM modulations. Four different values can be set for four TX power ranges.

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

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

set(*offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:IBE:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.qam.ibe.iqOffset.set(offset_0 = 1.0,␣
↪offset_1 = 1.0, offset_2 = 1.0, offset_3 = 1.0, qam = repcap.Qam.Default)
```

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission, for QAM modulations. Four different values can be set for four TX power ranges.

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

6.1.1.6.5.33 IqOffset

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>: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_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get(*qam=Qam.Default*) → IqOffsetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:IQOFfset
value: IqOffsetStruct = driver.configure.nrSubMeas.multiEval.limit.qam.iqOffset.
↪get(qam = repcap.Qam.Default)
```

Defines upper limits for the I/Q origin offset (QAM modulations) . Four different I/Q origin offset limits can be set for four TX power ranges.

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

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

set(*enable: bool*, *offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.qam.iqOffset.set(enable = False,␣
↪offset_0 = 1.0, offset_1 = 1.0, offset_2 = 1.0, offset_3 = 1.0, qam = repcap.
↪Qam.Default)
```

Defines upper limits for the I/Q origin offset (QAM modulations) . Four different I/Q origin offset limits can be set for four TX power ranges.

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

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

6.1.1.6.5.34 Merror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>: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(*qam=Qam.Default*) → MerrorStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:MERRor
value: MerrorStruct = driver.configure.nrSubMeas.multiEval.limit.qam.merror.
↪get(qam = repcap.Qam.Default)
```

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

param qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:MERRor
driver.configure.nrSubMeas.multiEval.limit.qam.merror.set(rms = 1.0, peak = 1.0,
↪ qam = repcap.Qam.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 qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

6.1.1.6.5.35 Perror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>: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(*qam=Qam.Default*) → PerrorStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:PERRor
value: PerrorStruct = driver.configure.nrSubMeas.multiEval.limit.qam.perror.
↪get(qam = repcap.Qam.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 qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

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

set(*rms: float*, *peak: float*, *qam=Qam.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QAM<order>
↪:PERRor
driver.configure.nrSubMeas.multiEval.limit.qam.perror.set(rms = 1.0, peak = 1.0,
↪ qam = repcap.Qam.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 qam
optional repeated capability selector. Default value: Order16 (settable in the interface 'Qam')

6.1.1.6.5.36 Qpsk

SCPI Commands :

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

class QpskCls

Qpsk commands group definition. 8 total commands, 5 Subgroups, 2 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: Band edge distance of border between range 1 and range 2

get_es_flatness() → EsFlatnessStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
value: EsFlatnessStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
value: float or bool = driver.configure.nrSubMeas.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

set_es_flatness(*value: EsFlatnessStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:ESFLatness
structure = driver.configure.nrSubMeas.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.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:FERRor
driver.configure.nrSubMeas.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

Cloning the Group

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

Subgroups

6.1.1.6.5.37 EvMagnitude

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
value: EvMagnitudeStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:EVMagnitude
driver.configure.nrSubMeas.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

6.1.1.6.5.38 Ibe

SCPI Command :

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

class IbeCls

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

class ValueStruct

Structure for setting input parameters. 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_Lesser: float: I/Q image for low TX power range
- Iq_Image_Greater: float: I/Q image for high TX power range

get_value() → ValueStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
value: ValueStruct = driver.configure.nrSubMeas.multiEval.limit.qpsk.ibe.get_
↪value()
```

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 ValueStruct structure arguments.

set_value(*value: ValueStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:IBE
structure = driver.configure.nrSubMeas.multiEval.limit.qpsk.ibe.ValueStruct()
structure.Enable: bool = False
structure.Minimum: float = 1.0
structure.Evm: float = 1.0
structure.Rb_Power: float = 1.0
structure.Iq_Image_Lesser: float = 1.0
structure.Iq_Image_Greater: float = 1.0
driver.configure.nrSubMeas.multiEval.limit.qpsk.ibe.set_value(value = structure)
```

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

param value
see the help for ValueStruct structure arguments.

Cloning the Group

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

Subgroups

6.1.1.6.5.39 IqOffset

SCPI Command :

```
CONFigure:NRSub: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_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get() → IqOffsetStruct

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

Defines I/Q origin offset values used for calculation of an upper limit for the in-band emission, for QPSK modulation. Four different values can be set for four TX power ranges.

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

set(*offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:QPSK:IBE:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.qpsk.ibe.iqOffset.set(offset_0 = 1.0,
↪ 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, for QPSK modulation. Four different values can be set for four TX power ranges.

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

6.1.1.6.5.40 IqOffset

SCPI Command :

```
CONFigure:NRSub: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_0: float: I/Q origin offset limit for TX power 10 dBm
- Offset_1: float: I/Q origin offset limit for TX power 0 dBm
- Offset_2: float: I/Q origin offset limit for TX power -30 dBm
- Offset_3: float: I/Q origin offset limit for TX power -40 dBm

get() → IqOffsetStruct

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

Defines upper limits for the I/Q origin offset (QPSK modulation) . Four different I/Q origin offset limits can be set for four TX power ranges.

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

set(*enable: bool*, *offset_0: float*, *offset_1: float*, *offset_2: float*, *offset_3: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:IQOFfset
driver.configure.nrSubMeas.multiEval.limit.qpsk.iqOffset.set(enable = False,␣
↪offset_0 = 1.0, offset_1 = 1.0, offset_2 = 1.0, offset_3 = 1.0)
```

Defines upper limits for the I/Q origin offset (QPSK modulation) . Four different I/Q origin offset limits can be set for four TX power ranges.

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

param offset_0
I/Q origin offset limit for TX power 10 dBm

param offset_1
I/Q origin offset limit for TX power 0 dBm

param offset_2
I/Q origin offset limit for TX power -30 dBm

param offset_3
I/Q origin offset limit for TX power -40 dBm

6.1.1.6.5.41 Merror

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
value: MerrorStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:MERRor
driver.configure.nrSubMeas.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

6.1.1.6.5.42 Perror

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
value: PerrorStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIMit:QPSK:PERRor
driver.configure.nrSubMeas.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

6.1.1.6.5.43 SeMask

class SeMaskCls

SeMask commands group definition. 14 total commands, 6 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.6.5.44 ActLimit

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:ACTLimit:ENDC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:ACTLimit:CAGGregation
```

class ActLimitCls

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

get_caggregation() → MevLimit

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:ACTLimit:CAGGregation
value: enums.MevLimit = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪actLimit.get_caggregation()
```

Selects the active spectrum emission mask and OBW limit for NR SA with carrier aggregation.

return
limit: STD: use standard limits UDEF: use user-defined limits

get_endc() → MevLimit

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:ACTLimit:ENDC
value: enums.MevLimit = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪actLimit.get_endc()
```

Selects the active spectrum emission mask and OBW limit for EN-DC.

return
limit: STD: use standard limits UDEF: use user-defined limits

set_caggregation(*limit: MevLimit*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:ACTLimit:CAGGregation
driver.configure.nrSubMeas.multiEval.limit.seMask.actLimit.set_
↪caggregation(limit = enums.MevLimit.STD)
```

Selects the active spectrum emission mask and OBW limit for NR SA with carrier aggregation.

param limit
STD: use standard limits UDEF: use user-defined limits

set_endc(*limit: MevLimit*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:ACTLimit:ENDC
driver.configure.nrSubMeas.multiEval.limit.seMask.actLimit.set_endc(limit =␣
↪enums.MevLimit.STD)
```

Selects the active spectrum emission mask and OBW limit for EN-DC.

param limit
STD: use standard limits UDEF: use user-defined limits

6.1.1.6.5.45 Area<Area>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.configure.nrSubMeas.multiEval.limit.seMask.area.repcap_area_get()
driver.configure.nrSubMeas.multiEval.limit.seMask.area.repcap_area_set(repcap.Area.Nr1)
```

class AreaCls

Area commands group definition. 3 total commands, 3 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.configure.nrSubMeas.multiEval.limit.seMask.area.clone()
```

Subgroups

6.1.1.6.5.46 Additional<AddTable>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.nrSubMeas.multiEval.limit.seMask.area.additional.repcap_addTable_
↪get()
driver.configure.nrSubMeas.multiEval.limit.seMask.area.additional.repcap_addTable_
↪set(repcap.AddTable.Nr1)
```

class AdditionalCls

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

Cloning the Group

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

Subgroups

6.1.1.6.5.47 Cbandwidth<ChannelBw>

RepCap Settings

```
# Range: Bw5 .. Bw100
rc = driver.configure.nrSubMeas.multiEval.limit.seMask.area.additional.cbandwidth.repcap_
↪channelBw_get()
driver.configure.nrSubMeas.multiEval.limit.seMask.area.additional.cbandwidth.repcap_
↪channelBw_set(repcap.ChannelBw.Bw5)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA<area>:ADDitional
↪<table>:CBANdwidth<bw>
```

class CbandwidthCls

Cbandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw5

class CbandwidthStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Stop frequency of the area, relative to the edges of the channel bandwidth.
- Frequency_End: float: No parameter help available
- Level: float: Upper limit for the area
- Rbw: enums.RbwB: Resolution bandwidth to be used for the area. Only a subset of the values is allowed, depending on table and bw. K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel bandwidth PC2: 2 % of channel bandwidth

get(*area=Area.Default*, *addTable=AddTable.Default*, *channelBw=ChannelBw.Default*) → CbandwidthStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA
↪<area>:ADDitional<table>:CBANdwidth<bw>
value: CbandwidthStruct = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪area.additional.cbandwidth.get(area = repcap.Area.Default, addTable = repcap.
↪AddTable.Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area number <area> (for NR SA without CA) . The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <bw>. Several tables of additional requirements are available.

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

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

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: RbwB*, *area=Area.Default*, *addTable=AddTable.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA
↪<area>:ADDitional<table>:CBANdwidth<bw>
driver.configure.nrSubMeas.multiEval.limit.seMask.area.additional.cbandwidth.
↪set(enable = False, frequency_start = 1.0, frequency_end = 1.0, level = 1.0,␣
↪rbw = enums.RbwB.K030, area = repcap.Area.Default, addTable = repcap.AddTable.
↪Default, channelBw = repcap.ChannelBw.Default)
```

Defines additional requirements for the emission mask area number <area> (for NR SA without CA) . The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <bw>. 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
Stop frequency of the area, relative to the edges of the channel bandwidth.

param frequency_end
No help available

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 bw. K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel bandwidth PC2: 2 % of channel bandwidth

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

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

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

Cloning the Group

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

6.1.1.6.5.48 Cbandwidth<ChannelBw>

RepCap Settings

```
# Range: Bw5 .. Bw100
rc = driver.configure.nrSubMeas.multiEval.limit.seMask.area.cbandwidth.repcap_channelBw_
↪get()
driver.configure.nrSubMeas.multiEval.limit.seMask.area.cbandwidth.repcap_channelBw_
↪set(repcap.ChannelBw.Bw5)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA<area>:CBANdwidth<bw>
```

class CbandwidthCls

Cbandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw5

class CbandwidthStruct

Response structure. Fields:

- Enable: bool: OFF: Disables the check of these requirements. ON: Enables the check of these requirements.
- Frequency_Start: float: Stop frequency of the area, relative to the edges of the channel bandwidth.
- Frequency_End: float: No parameter help available
- Level: float: Upper limit for the area
- Rbw: enums.RbwA: Resolution bandwidth to be used for the area Only a subset of the values is allowed, depending on bw. K030: 30 kHz PC1: 1 % of channel bandwidth M1: 1 MHz

get(*area=Area.Default*, *channelBw=ChannelBw.Default*) → CbandwidthStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA
↪<area>:CBANdwidth<bw>
value: CbandwidthStruct = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪area.cbandwidth.get(area = repcap.Area.Default, channelBw = repcap.ChannelBw.
↪Default)
```

Defines general requirements for the emission mask area number <area> (for NR SA without CA) . The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <bw>.

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

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: RbwA*, *area=Area.Default*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA
↪<area>:CBANdwidth<bw>
driver.configure.nrSubMeas.multiEval.limit.seMask.area.cbandwidth.set(enable =␣
↪False, frequency_start = 1.0, frequency_end = 1.0, level = 1.0, rbw = enums.
↪RbwA.K030, area = repcap.Area.Default, channelBw = repcap.ChannelBw.Default)
```

Defines general requirements for the emission mask area number <area> (for NR SA without CA) . The activation state, the area borders, an upper limit and the resolution bandwidth must be specified. The emission mask applies to the channel bandwidth <bw>.

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

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

param frequency_end
No help available

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 bw. K030: 30 kHz PC1: 1 % of channel bandwidth M1: 1 MHz

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

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

Cloning the Group

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

6.1.1.6.5.49 Endc

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA<nr>:ENDC
```

class EndcCls

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

class EndcStruct

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.RbwA: Resolution bandwidth to be used for the area. K030: 30 kHz PC1: 1 % of aggregated channel bandwidth M1: 1 MHz

get(*area=Area.Default*) → EndcStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA<nr>
↪:ENDC
value: EndcStruct = driver.configure.nrSubMeas.multiEval.limit.seMask.area.endc.
↪get(area = repcap.Area.Default)
```

Defines general requirements for area number <no> of the user-defined emission mask for EN-DC. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified.

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

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: RbwA*, *area=Area.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:AREA<nr>
↪:ENDC
driver.configure.nrSubMeas.multiEval.limit.seMask.area.endc.set(enable = False,␣
↪frequency_start = 1.0, frequency_end = 1.0, level = 1.0, rbw = enums.RbwA.
↪K030, area = repcap.Area.Default)
```

Defines general requirements for area number <no> of the user-defined emission mask for EN-DC. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified.

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 PC1: 1 % of aggregated channel bandwidth M1: 1 MHz

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

6.1.1.6.5.50 ObwLimit

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:OBWLimit:ENDC
```

class ObwLimitCls

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

get_endc() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:ENDC
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪obwLimit.get_endc()
```

Defines an upper user-defined limit for the OBW, for EN-DC measurements.

return
obw_limit: (float or boolean) No help available

set_endc(*obw_limit: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:ENDC
driver.configure.nrSubMeas.multiEval.limit.seMask.obwLimit.set_endc(obw_limit =␣
↪1.0)
```

Defines an upper user-defined limit for the OBW, for EN-DC measurements.

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.nrSubMeas.multiEval.limit.seMask.obwLimit.clone()
```

Subgroups

6.1.1.6.5.51 Cbandwidth<ChannelBw>

RepCap Settings

```
# Range: Bw5 .. Bw100
rc = driver.configure.nrSubMeas.multiEval.limit.seMask.obwLimit.cbandwidth.repcap_
↪channelBw_get()
driver.configure.nrSubMeas.multiEval.limit.seMask.obwLimit.cbandwidth.repcap_channelBw_
↪set(repcap.ChannelBw.Bw5)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<bw>
```

class CbandwidthCls

Cbandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw5

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<bw>
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪obwLimit.cbandwidth.get(channelBw = repcap.ChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth, depending on the channel bandwidth (for NR SA without CA) .

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

return
obw_limit: (float or boolean) No help available

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:OBWLimit:CBANdwidth<bw>
driver.configure.nrSubMeas.multiEval.limit.seMask.obwLimit.cbandwidth.set(obw_
↪limit = 1.0, channelBw = repcap.ChannelBw.Default)
```

Defines an upper limit for the occupied bandwidth, depending on the channel bandwidth (for NR SA without CA) .

param obw_limit
(float or boolean) No help available

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

Cloning the Group

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

6.1.1.6.5.52 Standard

class StandardCls

Standard 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.nrSubMeas.multiEval.limit.seMask.standard.clone()
```

Subgroups

6.1.1.6.5.53 Area<AreaReduced>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.nrSubMeas.multiEval.limit.seMask.standard.area.repcap_areaReduced_
↪get()
driver.configure.nrSubMeas.multiEval.limit.seMask.standard.area.repcap_areaReduced_
↪set(repcap.AreaReduced.Nr1)
```

class AreaCls

Area commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: AreaReduced, default value after init: AreaReduced.Nr1

Cloning the Group

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

Subgroups

6.1.1.6.5.54 Caggregation

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:STANdard:AREA<nr>
↪:CAGGregation
```

class CaggregationCls

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

get(*areaReduced=AreaReduced.Default*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:AREA<nr>:CAGGregation
value: bool = driver.configure.nrSubMeas.multiEval.limit.seMask.standard.area.
↪caggregation.get(areaReduced = repcap.AreaReduced.Default)
```

Configures the activation state of area number <no> of the standard emission mask for NR SA with carrier aggregation.

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

return
enable: OFF: disables the limit check for this area ON: enables the limit check for this area

set(*enable: bool*, *areaReduced=AreaReduced.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:AREA<nr>:CAGGregation
driver.configure.nrSubMeas.multiEval.limit.seMask.standard.area.caggregation.
↪set(enable = False, areaReduced = repcap.AreaReduced.Default)
```

Configures the activation state of area number <no> of the standard emission mask for NR SA with carrier aggregation.

param enable
OFF: disables the limit check for this area ON: enables the limit check for this area

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

6.1.1.6.5.55 Endc

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:STANdard:AREA<nr>:ENDC
```

class EndcCls

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

get(*areaReduced=AreaReduced.Default*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:AREA<nr>:ENDC
value: bool = driver.configure.nrSubMeas.multiEval.limit.seMask.standard.area.
↪endc.get(areaReduced = repcap.AreaReduced.Default)
```

Configures the activation state of area number <no> of the standard emission mask for EN-DC.

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

return
enable: OFF: disables the limit check for this area ON: enables the limit check for this area

set(*enable: bool*, *areaReduced=AreaReduced.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:AREA<nr>:ENDC
driver.configure.nrSubMeas.multiEval.limit.seMask.standard.area.endc.set(enable␣
↪= False, areaReduced = repcap.AreaReduced.Default)
```

Configures the activation state of area number <no> of the standard emission mask for EN-DC.

param enable
OFF: disables the limit check for this area ON: enables the limit check for this area

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

6.1.1.6.5.56 ObwLimit

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:STANdard:OBWLimit:ENDC
CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:OBWLimit:CAGGregation
```

class ObwLimitCls

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

get_caggregation() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:OBWLimit:CAGGregation
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪standard.obwLimit.get_caggregation()
```

Configures the activation state of the standard OBW limit for carrier aggregation measurements.

return
obw_limit: (float or boolean) A setting allows only ON | OFF. A query returns the limit value instead of ON (numeric | OFF) .

get_endc() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:OBWLimit:ENDC
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪standard.obwLimit.get_endc()
```

Configures the activation state of the standard OBW limit for EN-DC measurements.

return
obw_limit: (float or boolean) A setting allows only ON | OFF. A query returns the limit value instead of ON (numeric | OFF) .

set_caggregation(*obw_limit: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:OBWLimit:CAGGregation
driver.configure.nrSubMeas.multiEval.limit.seMask.standard.obwLimit.set_
↪caggregation(obw_limit = 1.0)
```

Configures the activation state of the standard OBW limit for carrier aggregation measurements.

param obw_limit
(float or boolean) A setting allows only ON | OFF. A query returns the limit value instead of ON (numeric | OFF) .

set_endc(*obw_limit: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:STANdard:OBWLimit:ENDC
driver.configure.nrSubMeas.multiEval.limit.seMask.standard.obwLimit.set_
↪endc(obw_limit = 1.0)
```

Configures the activation state of the standard OBW limit for EN-DC measurements.

param obw_limit
(float or boolean) A setting allows only ON | OFF. A query returns the limit value instead of ON (numeric | OFF) .

6.1.1.6.5.57 Ttolerance

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:TTOLerance
```

class TtoleranceCls

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

class TtoleranceStruct

Response structure. Fields:

- Test_Tol_Sub_3_Ghz: float: Test tolerance for center frequencies 3 GHz
- Test_Tol_Sub_4_Ghz: float: Test tolerance for center frequencies 3 GHz and 4.2 GHz
- Test_Tol_Sub_6_Gh_Z: float: Test tolerance for center frequencies 4.2 GHz

get() → TtoleranceStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:TTOLerance
value: TtoleranceStruct = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪ttolerance.get()
```

Defines the test tolerance for spectrum emission masks, depending on the center frequency.

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

set(*test_tol_sub_3_ghz: float*, *test_tol_sub_4_ghz: float*, *test_tol_sub_6_gh_z: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:TTOLerance
driver.configure.nrSubMeas.multiEval.limit.seMask.ttolerance.set(test_tol_sub_3_
↪ghz = 1.0, test_tol_sub_4_ghz = 1.0, test_tol_sub_6_gh_z = 1.0)
```

Defines the test tolerance for spectrum emission masks, depending on the center frequency.

param test_tol_sub_3_ghz
Test tolerance for center frequencies 3 GHz

param test_tol_sub_4_ghz
Test tolerance for center frequencies 3 GHz and 4.2 GHz

param test_tol_sub_6_gh_z
Test tolerance for center frequencies 4.2 GHz

6.1.1.6.5.58 UserDefined

class UserDefinedCls

UserDefined 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.nrSubMeas.multiEval.limit.seMask.userDefined.clone()
```

Subgroups

6.1.1.6.5.59 Area<Area>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.configure.nrSubMeas.multiEval.limit.seMask.userDefined.area.repcap_area_get()
driver.configure.nrSubMeas.multiEval.limit.seMask.userDefined.area.repcap_area_
↪set(repcap.Area.Nr1)
```

class AreaCls

Area commands group definition. 1 total commands, 1 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.configure.nrSubMeas.multiEval.limit.seMask.userDefined.area.clone()
```

Subgroups

6.1.1.6.5.60 Caggregation

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SEMask:UDEFined:AREA<nr>
↪:CAGGregation
```

class CaggregationCls

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

class CaggregationStruct

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.RbwC: Resolution bandwidth to be used for the area K030: 30 kHz K400: 400 kHz M1: 1 MHz

get(*area=Area.Default*) → CaggregationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:UDEFined:AREA<nr>:CAGGregation
value: CaggregationStruct = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪userDefined.area.caggregation.get(area = repcap.Area.Default)
```

Defines general requirements for area number <no> of the user-defined emission mask for NR SA with carrier aggregation. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified.

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

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

set(*enable: bool*, *frequency_start: float*, *frequency_end: float*, *level: float*, *rbw: RbwC*, *area=Area.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:UDEFined:AREA<nr>:CAGGregation
driver.configure.nrSubMeas.multiEval.limit.seMask.userDefined.area.caggregation.
↪set(enable = False, frequency_start = 1.0, frequency_end = 1.0, level = 1.0,␣
↪rbw = enums.RbwC.K030, area = repcap.Area.Default)
```

Defines general requirements for area number <no> of the user-defined emission mask for NR SA with carrier aggregation. The activation state, the area borders, an upper limit and the resolution bandwidth must be specified.

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 K400: 400 kHz M1: 1 MHz

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

6.1.1.6.5.61 ObwLimit

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:UDEFined:OBWLimit:CAGGregation
```

class ObwLimitCls

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

get_caggregation() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:UDEFined:OBWLimit:CAGGregation
value: float or bool = driver.configure.nrSubMeas.multiEval.limit.seMask.
↪userDefined.obwLimit.get_caggregation()
```

Defines an upper user-defined limit for the OBW, for carrier aggregation measurements.

return
obw_limit: (float or boolean) No help available

set_caggregation(*obw_limit: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:LIMit:SEMask:UDEFined:OBWLimit:CAGGregation
driver.configure.nrSubMeas.multiEval.limit.seMask.userDefined.obwLimit.set_
↪caggregation(obw_limit = 1.0)
```

Defines an upper user-defined limit for the OBW, for carrier aggregation measurements.

param obw_limit
(float or boolean) No help available

6.1.1.6.6 ListPy

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:OSINdex
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:CMODe
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:NCONnections
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST
```

class ListPyCls

ListPy commands group definition. 17 total commands, 3 Subgroups, 4 group commands

get_cmode() → ParameterSetMode

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:CMODe
value: enums.ParameterSetMode = driver.configure.nrSubMeas.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:NRSub:MEASi:SPATh. - LIST: Assign a connection to each segment, see CONFigure:NRSub:MEASi:MEValuation:LIST:SEGMentno:CIDX.

get_nconnections() → int

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

No command help available

return
offline_seg_index: (integer or boolean) No help available

get_value() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST
value: bool = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:CMODe
driver.configure.nrSubMeas.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:NRSub:MEASi:SPATh.
- LIST: Assign a connection to each segment, see CONFigure:NRSub:MEASi:MEValuation:LIST:SEGMentno:CIDX.

set_nconnections(*no_of_connections: int*) → None

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

No command help available

param offline_seg_index
(integer or boolean) No help available

set_value(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST
driver.configure.nrSubMeas.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.nrSubMeas.multiEval.listPy.clone()
```

Subgroups

6.1.1.6.6.1 Lrange

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIST:LRANge
value: LrangeStruct = driver.configure.nrSubMeas.multiEval.listPy.lrange.get()
```

Select a range of measured segments.

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

set(*start_index: int*, *nr_segments: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:LRANge
driver.configure.nrSubMeas.multiEval.listPy.lrange.set(start_index = 1, nr_
↪segments = 1)
```

Select a range of measured segments.

param start_index
First measured segment in the range of configured segments

param nr_segments
Number of measured segments

6.1.1.6.6.2 Segment

RepCap Settings

```
# Range: Nr1 .. Nr512
rc = driver.configure.nrSubMeas.multiEval.listPy.segment.repcap_sEGMent_get()
driver.configure.nrSubMeas.multiEval.listPy.segment.repcap_sEGMent_set(repcap.SEGMent.
↪Nr1)
```

class SegmentCls

Segment commands group definition. 11 total commands, 11 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.nrSubMeas.multiEval.listPy.segment.clone()
```

Subgroups

6.1.1.6.6.3 Aclr

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:ACLR
```

class AclrCls

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

class AclrStruct

Structure for setting input parameters. Contains optional setting parameters. 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. the power in the assigned NR 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.
- Nr_Enable: bool: Enable or disable evaluation of first adjacent NR channels.
- Endc_Enable: bool: Optional setting parameter. Enable or disable evaluation of adjacent channel power in EN-DC mode.

get(*sEGMent=SEGMent.Default*) → AclrStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:ACLR
value: AclrStruct = driver.configure.nrSubMeas.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(*structure: AclrStruct*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:ACLR
structure = driver.configure.nrSubMeas.multiEval.listPy.segment.aclr.
↪AclrStruct()
structure.Aclr_Statistics: int = 1
structure.Aclr_Enable: bool = False
structure.Utra_1_Enable: bool = False
structure.Utra_2_Enable: bool = False
structure.Nr_Enable: bool = False
structure.Endc_Enable: bool = False
driver.configure.nrSubMeas.multiEval.listPy.segment.aclr.set(structure, sEGMent␣
↪= repcap.SEGMent.Default)
```

Defines settings for ACLR measurements in list mode for segment <no>.

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

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

6.1.1.6.6.4 Cidx

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:CIDX
```

class CidxCls

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

get(*sEGMent=SEGMent.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:CIDX
value: int = driver.configure.nrSubMeas.multiEval.listPy.segment.cidx.
↪get(sEGMent = repcap.SEGMent.Default)
```

Selects the RF connection index for segment <no>. For a definition of the connection indices, see ROUTe:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:CIDX
driver.configure.nrSubMeas.multiEval.listPy.segment.cidx.set(connection_index =␣
↪1, sEGMent = repcap.SEGMent.Default)
```

Selects the RF connection index for segment <no>. For a definition of the connection indices, see ROUTe:NRSub:MEAS<i>:SPATh.

param connection_index
No help available

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

6.1.1.6.6.5 Endc

class EndcCls

Endc 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.nrSubMeas.multiEval.listPy.segment.endc.clone()
```

Subgroups

6.1.1.6.6.6 Eutra

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:ENDC:EUTRa
```

class EutraCls

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

class EutraStruct

Response structure. Fields:

- Channel_Bw: enums.ChannelBwidthB: Channel bandwidth in MHz (5 MHz to 20 MHz) .
- Carrier_Position: enums.CarrierPosition: Position of LTE carrier left (LONR) or right (RONR) of NR carrier.

get(*sEGMent=SEGMent.Default*) → EutraStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:ENDC:EUTRa
value: EutraStruct = driver.configure.nrSubMeas.multiEval.listPy.segment.endc.
↪eutra.get(sEGMent = repcap.SEGMent.Default)
```

Configures LTE settings for the EN-DC 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 EutraStruct structure arguments.

set(*channel_bw: ChannelBwidthB*, *carrier_position: CarrierPosition*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:ENDC:EUTRa
driver.configure.nrSubMeas.multiEval.listPy.segment.endc.eutra.set(channel_bw =␣
↪enums.ChannelBwidthB.B005, carrier_position = enums.CarrierPosition.LONR,␣
↪sEGMent = repcap.SEGMent.Default)
```

Configures LTE settings for the EN-DC mode, for segment <no>.

param channel_bw
Channel bandwidth in MHz (5 MHz to 20 MHz) .

param carrier_position
Position of LTE carrier left (LONR) or right (RONR) of NR carrier.

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

6.1.1.6.6.7 Fdistance

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:FDIStance
```

class FdistanceCls

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

class GetStruct

Response structure. Fields:

- Left: float: No parameter help available
- Right: float: No parameter help available

get(*sEGMent=SEGMent.Default*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:FDIStance
value: GetStruct = driver.configure.nrSubMeas.multiEval.listPy.segment.
↪fdistance.get(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 GetStruct structure arguments.

6.1.1.6.6.8 Modulation

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>: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.

get(*sEGMent=SEGMent.Default*) → ModulationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:MODulation
value: ModulationStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:MODulation
structure = driver.configure.nrSubMeas.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
structure.Phase_Err_Enable: bool = False
structure.Ib_Eenable: bool = False
structure.Eq_Sp_Flat_Enable: bool = False
driver.configure.nrSubMeas.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')

6.1.1.6.6.9 Pmonitor

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:PMONitor
```

class PmonitorCls

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

get(*sEGMent=SEGMent.Default*) → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:PMONitor
value: bool = driver.configure.nrSubMeas.multiEval.listPy.segment.pmonitor.
↪get(sEGMent = repcap.SEGMent.Default)
```

Enables or disables the measurement of power monitor results 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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:PMONitor
driver.configure.nrSubMeas.multiEval.listPy.segment.pmonitor.set(enable = False,
↪ sEGMent = repcap.SEGMent.Default)
```

Enables or disables the measurement of power monitor results for segment <no>.

param enable
No help available

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

6.1.1.6.6.10 Power

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>: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_Tx_Enable: bool: Enables or disables the measurement of the total TX power.

get(*sEGMent=SEGMent.Default*) → PowerStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:POWer
value: PowerStruct = driver.configure.nrSubMeas.multiEval.listPy.segment.power.
↪get(sEGMent = repcap.SEGMent.Default)
```

Defines settings for the measurement of the total TX power 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_tx_enable: bool*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:POWer
driver.configure.nrSubMeas.multiEval.listPy.segment.power.set(power_statistics␣
↪= 1, power_tx_enable = False, sEGMent = repcap.SEGMent.Default)
```

Defines settings for the measurement of the total TX power for segment <no>.

param power_statistics
Statistical length in subframes

param power_tx_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')

6.1.1.6.6.11 PuschConfig

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:PUSChconfig
```

class PuschConfigCls

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

class PuschConfigStruct

Structure for setting input parameters. Fields:

- Mod_Scheme: enums.ModulationScheme: No parameter help available
- Mapping_Type: enums.MappingType: No parameter help available
- Nrb_Auto: bool: No parameter help available
- No_Rb: int: No parameter help available
- Start_Rb: int: No parameter help available
- No_Symbols: int: No parameter help available
- Start_Symbol: int: No parameter help available
- Config_Type: enums.ConfigType: No parameter help available

- Max_Length: enums.MaxLength: No parameter help available
- Add_Position: int: No parameter help available
- Lzero: int: No parameter help available

get(*sEGMent=SEGMent.Default*) → PuschConfigStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:PUSChconfig
value: PuschConfigStruct = driver.configure.nrSubMeas.multiEval.listPy.segment.
↪puschConfig.get(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 PuschConfigStruct structure arguments.

set(*structure: PuschConfigStruct*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:PUSChconfig
structure = driver.configure.nrSubMeas.multiEval.listPy.segment.puschConfig.
↪PuschConfigStruct()
structure.Mod_Scheme: enums.ModulationScheme = enums.ModulationScheme.AUTO
structure.Mapping_Type: enums.MappingType = enums.MappingType.A
structure.Nrb_Auto: bool = False
structure.No_Rb: int = 1
structure.Start_Rb: int = 1
structure.No_Symbols: int = 1
structure.Start_Symbol: int = 1
structure.Config_Type: enums.ConfigType = enums.ConfigType.T1
structure.Max_Length: enums.MaxLength = enums.MaxLength.DOUBle
structure.Add_Position: int = 1
structure.Lzero: int = 1
driver.configure.nrSubMeas.multiEval.listPy.segment.puschConfig.set(structure,␣
↪sEGMent = repcap.SEGMent.Default)
```

No command help available

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

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

6.1.1.6.6.12 SeMask

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:SEMask
value: SeMaskStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:SEMask
driver.configure.nrSubMeas.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')

6.1.1.6.6.13 Setup

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>: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: No parameter help available
- Level: float: No parameter help available
- Duplex_Mode: enums.DuplexModeB: No parameter help available
- Band: enums.Band: No parameter help available
- Frequency: float: No parameter help available
- Sub_Carr_Spacing: enums.SubCarrSpacing: No parameter help available
- Ch_Bandwidth: enums.ChannelBwidth: No parameter help available
- Cyclic_Prefix: enums.CyclicPrefix: No parameter help available
- Channel_Type: enums.ChannelTypeA: No parameter help available
- Dft_Precoding: bool: No parameter help available
- Retrigger_Flag: enums.RetriggerFlag: No parameter help available
- Evaluat_Offset: int: No parameter help available
- Network_Sig_Val: enums.NetworkSigVal: No parameter help available

`get`(*sEGMent=SEGMent.Default*) → SetupStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:SETup
value: SetupStruct = driver.configure.nrSubMeas.multiEval.listPy.segment.setup.
↪get(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 SetupStruct structure arguments.

set(*structure: SetupStruct*, *sEGMent=SEGMent.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:SETup
structure = driver.configure.nrSubMeas.multiEval.listPy.segment.setup.
↪SetupStruct()
structure.Segment_Length: int = 1
structure.Level: float = 1.0
structure.Duplex_Mode: enums.DuplexModeB = enums.DuplexModeB.FDD
structure.Band: enums.Band = enums.Band.OB1
structure.Frequency: float = 1.0
structure.Sub_Carr_Spacing: enums.SubCarrSpacing = enums.SubCarrSpacing.S15K
structure.Ch_Bandwidth: enums.ChannelBwidth = enums.ChannelBwidth.B005
structure.Cyclic_Prefix: enums.CyclicPrefix = enums.CyclicPrefix.EXTended
structure.Channel_Type: enums.ChannelTypeA = enums.ChannelTypeA.PUCCh
structure.Dft_Precoding: bool = False
structure.Retrigger_Flag: enums.RetriggerFlag = enums.RetriggerFlag.
↪IFPNarrowband
structure.Evaluat_Offset: int = 1
structure.Network_Sig_Val: enums.NetworkSigVal = enums.NetworkSigVal.NS01
driver.configure.nrSubMeas.multiEval.listPy.segment.setup.set(structure,␣
↪sEGMent = repcap.SEGMent.Default)
```

No command help available

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')

6.1.1.6.6.14 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.nrSubMeas.multiEval.listPy.segment.singleCmw.clone()
```

Subgroups

6.1.1.6.6.15 Connector

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>:CMWS:CONNector
```

class ConnectorCls

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

get(*sEGMent=SEGMent.Default*) → CmwsConnector

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:CMWS:CONNector
value: enums.CmwsConnector = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<no>
↪:CMWS:CONNector
driver.configure.nrSubMeas.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')

6.1.1.6.6.16 SingleCmw

SCPI Command :

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

class SingleCmwCls

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

get_cmode() → ParameterSetMode

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

No command help available

return
connector_mode: No help available

set_cmode(*connector_mode: ParameterSetMode*) → None

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

No command help available

param connector_mode
No help available

6.1.1.6.7 Modulation

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:TDLoffset
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:DAReceiver
```

class ModulationCls

Modulation commands group definition. 9 total commands, 3 Subgroups, 2 group commands

get_da_receiver() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:DAReceiver
value: bool = driver.configure.nrSubMeas.multiEval.modulation.get_da_receiver()
```

Enables the dual antenna receiver for measurements with two antennas. This setting selects how the signals of the two antennas are processed for the calculation of modulation results provided per layer / antenna.

return
enable: - OFF: The results are derived per antenna. - ON: The results are derived per layer from the combination of both antenna signals.

get_tdl_offset() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:TDLoffset
value: float = driver.configure.nrSubMeas.multiEval.modulation.get_tdl_offset()
```

Specifies the offset of the DC subcarrier from the center frequency (number of subcarriers) .

return
offset: No help available

set_da_receiver(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:DAReceiver
driver.configure.nrSubMeas.multiEval.modulation.set_da_receiver(enable = False)
```

Enables the dual antenna receiver for measurements with two antennas. This setting selects how the signals of the two antennas are processed for the calculation of modulation results provided per layer / antenna.

param enable

- OFF: The results are derived per antenna.
- ON: The results are derived per layer from the combination of both antenna signals.

set_tdl_offset(*offset: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:TDLoffset
driver.configure.nrSubMeas.multiEval.modulation.set_tdl_offset(offset = 1.0)
```

Specifies the offset of the DC subcarrier from the center frequency (number of subcarriers) .

param offset
No help available

Cloning the Group

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

Subgroups

6.1.1.6.7.1 EePeriods

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUCCh
value: bool = driver.configure.nrSubMeas.multiEval.modulation.eePeriods.get_
↪pucch()
```

No command help available

return
pucch: No help available

set_pucch(*pucch: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUCCh
driver.configure.nrSubMeas.multiEval.modulation.eePeriods.set_pucch(pucch =␣
↪False)
```

No command help available

param pucch
No help available

Cloning the Group

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

Subgroups

6.1.1.6.7.2 Pusch

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
```

class PuschCls

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

get_lagging() → Lagging

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
value: enums.Lagging = driver.configure.nrSubMeas.multiEval.modulation.
↪eePeriods.pusch.get_lagging()
```

No command help available

return
lagging: No help available

get_leading() → Leading

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
value: enums.Leading = driver.configure.nrSubMeas.multiEval.modulation.
↪eePeriods.pusch.get_leading()
```

No command help available

return
leading: No help available

set_lagging(*lagging: Lagging*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LAGGing
driver.configure.nrSubMeas.multiEval.modulation.eePeriods.pusch.set_
↪lagging(lagging = enums.Lagging.MS05)
```

No command help available

param lagging
No help available

set_leading(*leading: Leading*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EEPeriods:PUSCh:LEADing
driver.configure.nrSubMeas.multiEval.modulation.eePeriods.pusch.set_
↪leading(leading = enums.Leading.MS25)
```

No command help available

param leading
No help available

6.1.1.6.7.3 EwLength

class EwLengthCls

EwLength 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.nrSubMeas.multiEval.modulation.ewLength.clone()
```

Subgroups

6.1.1.6.7.4 Cbandwidth<ChannelBw>

RepCap Settings

```
# Range: Bw5 .. Bw100
rc = driver.configure.nrSubMeas.multiEval.modulation.ewLength.cbandwidth.repcap_
↪channelBw_get()
driver.configure.nrSubMeas.multiEval.modulation.ewLength.cbandwidth.repcap_channelBw_
↪set(repcap.ChannelBw.Bw5)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:EWLength:CBANdwidth<bw>
```

class CbandwidthCls

Cbandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw5

class CbandwidthStruct

Response structure. Fields:

- Cyc_Prefix_Norm_15: int: Samples for normal CP, 15-kHz SC spacing
- Cyc_Prefix_Norm_30: int: Samples for normal CP, 60-kHz SC spacing
- Cyc_Prefix_Norm_60: int: Samples for extended CP, 60-kHz SC spacing
- Cyc_Prefix_Extend: int: No parameter help available

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EWLength:CBANdwidth<bw>
value: CbandwidthStruct = driver.configure.nrSubMeas.multiEval.modulation.
↪ewLength.cbandwidth.get(channelBw = repcap.ChannelBw.Default)
```

Specifies the EVM window length in samples for a selected channel bandwidth, depending on the cyclic prefix (CP) type and the SC spacing.

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

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

set(*cyc_prefix_norm_15: int*, *cyc_prefix_norm_30: int*, *cyc_prefix_norm_60: int*, *cyc_prefix_extend: int*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:EWLength:CBANdwidth<bw>
driver.configure.nrSubMeas.multiEval.modulation.ewLength.cbandwidth.set(cyc_
```

(continues on next page)

(continued from previous page)

```
↪prefix_norm_15 = 1, cyc_prefix_norm_30 = 1, cyc_prefix_norm_60 = 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 and the SC spacing.

param cyc_prefix_norm_15
Samples for normal CP, 15-kHz SC spacing

param cyc_prefix_norm_30
Samples for normal CP, 60-kHz SC spacing

param cyc_prefix_norm_60
Samples for extended CP, 60-kHz SC spacing

param cyc_prefix_extend
No help available

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

Cloning the Group

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

6.1.1.6.7.5 Tracking

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:TRACking:PHASe
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:TRACking:TIMing
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulation:TRACking:LEVel
```

class TrackingCls

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

get_level() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:TRACking:LEVel
value: bool = driver.configure.nrSubMeas.multiEval.modulation.tracking.get_
↪level()
```

Activate or deactivate level tracking. With enabled tracking, fluctuations are compensated.

return
level: OFF: Tracking disabled ON: Tracking enabled

get_phase() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:TRACking:PHASe
value: bool = driver.configure.nrSubMeas.multiEval.modulation.tracking.get_
↪phase()
```

Activate or deactivate phase tracking. With enabled tracking, fluctuations are compensated.

return
phase: OFF: Tracking disabled ON: Tracking enabled

get_timing() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:TRACking:TIMing
value: bool = driver.configure.nrSubMeas.multiEval.modulation.tracking.get_
↪timing()
```

Activate or deactivate timing tracking. With enabled tracking, fluctuations are compensated.

return
timing: OFF: Tracking disabled ON: Tracking enabled

set_level(*level: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:TRACking:LEVel
driver.configure.nrSubMeas.multiEval.modulation.tracking.set_level(level =␣
↪False)
```

Activate or deactivate level tracking. With enabled tracking, fluctuations are compensated.

param level
OFF: Tracking disabled ON: Tracking enabled

set_phase(*phase: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:TRACking:PHASe
driver.configure.nrSubMeas.multiEval.modulation.tracking.set_phase(phase =␣
↪False)
```

Activate or deactivate phase tracking. With enabled tracking, fluctuations are compensated.

param phase
OFF: Tracking disabled ON: Tracking enabled

set_timing(*timing: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:MODulation:TRACking:TIMing
driver.configure.nrSubMeas.multiEval.modulation.tracking.set_timing(timing =␣
↪False)
```

Activate or deactivate timing tracking. With enabled tracking, fluctuations are compensated.

param timing
OFF: Tracking disabled ON: Tracking enabled

6.1.1.6.8 Mslot

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MSLot
```

class MslotCls

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

class MslotStruct

Response structure. Fields:

- Measure_Slot: enums.MeasureSlot: ALL: all slots UDEF: single slot selected via MeasSlotNo
- Meas_Slot_No: int: Slot number for MeasureSlot=UDEF The number of slots per subframe depends on the SCS. The slot must be within the captured number of subframes, limited to 10 subframes, see [CMDLINKRESOLVED Configure.NrSubMeas.MultiEval#NsubFrames CMDLINKRESOLVED].

get() → MslotStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MSLot
value: MslotStruct = driver.configure.nrSubMeas.multiEval.mslot.get()
```

Selects which slots within the first 10 captured subframes are evaluated.

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

set(*measure_slot: MeasureSlot*, *meas_slot_no: int = None*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MSLot
driver.configure.nrSubMeas.multiEval.mslot.set(measure_slot = enums.MeasureSlot.
↪ALL, meas_slot_no = 1)
```

Selects which slots within the first 10 captured subframes are evaluated.

param measure_slot
ALL: all slots UDEF: single slot selected via MeasSlotNo

param meas_slot_no
Slot number for MeasureSlot=UDEF The number of slots per subframe depends on the SCS. The slot must be within the captured number of subframes, limited to 10 subframes, see method RsCMPX_NrFr1Meas.Configure.NrSubMeas.MultiEval.nsubFrames.

6.1.1.6.9 MsubFrames

SCPI Command :

```
CONFigure:NRSub: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: No parameter help available
- Sub_Frame_Count: int: No parameter help available
- Meas_Subframe: int: No parameter help available

get() → MsubFramesStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:MSUBframes
value: MsubFramesStruct = driver.configure.nrSubMeas.multiEval.msubFrames.get()
```

No command help available

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:NRSub:MEASurement<Instance>:MEValuation:MSUBframes
driver.configure.nrSubMeas.multiEval.msubFrames.set(sub_frame_offset = 1, sub_
↪frame_count = 1, meas_subframe = 1)
```

No command help available

param sub_frame_offset
No help available

param sub_frame_count
No help available

param meas_subframe
No help available

6.1.1.6.10 Pcomp

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:PCOMp
```

class PcompCls

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

class PcompStruct

Response structure. Fields:

- Phase_Comp: enums.PhaseComp: OFF: no phase compensation CAF: phase compensation for carrier frequency UDEF: phase compensation for frequency UserDefFreq
- User_Def_Freq: float or bool: Frequency for PhaseComp = UDEF

get() → PcompStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PCOMp
value: PcompStruct = driver.configure.nrSubMeas.multiEval.pcomp.get()
```

Specifies the phase compensation applied by the UE during the modulation and upconversion.

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

set(*phase_comp: PhaseComp*, *user_def_freq: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PCOMp
driver.configure.nrSubMeas.multiEval.pcomp.set(phase_comp = enums.PhaseComp.CAF,
↪ user_def_freq = 1.0)
```

Specifies the phase compensation applied by the UE during the modulation and upconversion.

param phase_comp
OFF: no phase compensation CAF: phase compensation for carrier frequency UDEF: phase compensation for frequency UserDefFreq

param user_def_freq
(float or boolean) Frequency for PhaseComp = UDEF

6.1.1.6.11 Pdynamics

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
CONFigure:NRSub:MEASurement<Instance>:MEValuation:PDYNamics:HDMode
```

class PdynamicsCls

Pdynamics commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_hdmode() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PDYNamics:HDMode
value: bool = driver.configure.nrSubMeas.multiEval.pdynamics.get_hdmode()
```

Enables or disables the high dynamic mode for power dynamics measurements.

return
high_dynamic_mode: No help available

get_tmask() → TimeMask

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
value: enums.TimeMask = driver.configure.nrSubMeas.multiEval.pdynamics.get_
↪tmask()
```

No command help available

return
time_mask: No help available

set_hdmode(*high_dynamic_mode: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PDYNamics:HDMode
driver.configure.nrSubMeas.multiEval.pdynamics.set_hdmode(high_dynamic_mode =␣
↪False)
```

Enables or disables the high dynamic mode for power dynamics measurements.

param high_dynamic_mode
No help available

set_tmask(*time_mask: TimeMask*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:PDYNamics:TMASk
driver.configure.nrSubMeas.multiEval.pdynamics.set_tmask(time_mask = enums.
↪TimeMask.GOO)
```

No command help available

param time_mask
No help available

Cloning the Group

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

Subgroups

6.1.1.6.11.1 AeoPower

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:PDYNamics:AEOPower:LEADing
CONFigure:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LAGGing
value: int = driver.configure.nrSubMeas.multiEval.pdynamics.aeoPower.get_
↪lagging()
```

No command help available

return
lagging: No help available

get_leading() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LEADing
value: int = driver.configure.nrSubMeas.multiEval.pdynamics.aeoPower.get_
↪leading()
```

No command help available

return
leading: No help available

set_lagging(*lagging: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LAGGing
driver.configure.nrSubMeas.multiEval.pdynamics.aeoPower.set_lagging(lagging = 1)
```

No command help available

param lagging
No help available

set_leading(*leading: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:PDYNamics:AEOPower:LEADing
driver.configure.nrSubMeas.multiEval.pdynamics.aeoPower.set_leading(leading = 1)
```

No command help available

param leading
No help available

6.1.1.6.12 RbAllocation

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:NRB
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:SRB
```

class RbAllocationCls

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

get_auto() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
value: bool = driver.configure.nrSubMeas.multiEval.rbAllocation.get_auto()
```

No command help available

return
auto: No help available

get_nrb() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:NRB
value: int = driver.configure.nrSubMeas.multiEval.rbAllocation.get_nrb()
```

No command help available

return
no_rb: No help available

get_srb() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:SRB
value: int = driver.configure.nrSubMeas.multiEval.rbAllocation.get_srb()
```

No command help available

return
start_rb: No help available

set_auto(*auto: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:AUTO
driver.configure.nrSubMeas.multiEval.rbAllocation.set_auto(auto = False)
```

No command help available

param auto
No help available

set_nrb(*no_rb: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:NRB
driver.configure.nrSubMeas.multiEval.rbAllocation.set_nrb(no_rb = 1)
```

No command help available

param no_rb
No help available

set_srb(*start_rb: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RBALlocation:SRB
driver.configure.nrSubMeas.multiEval.rbAllocation.set_srb(start_rb = 1)
```

No command help available

param start_rb
No help available

6.1.1.6.13 Result

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:MODulation
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:SEMask
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:ACLR
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:PMONitor
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:PDYNamics
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:TXPower
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult[:ALL]
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:MERRor
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:PERRor
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:IEMissions
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:EVMC
```

(continues on next page)

(continued from previous page)

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:IQ
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:ESFLatness
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:TXM
```

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: No parameter help available
- 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: No parameter help available
- Iq: bool: No parameter help available
- Equ_Spec_Flatness: bool: No parameter help available
- Tx_Measurement: bool: No parameter help available
- Spec_Em_Mask: bool: No parameter help available
- Aclr: bool: No parameter help available
- Power_Monitor: bool: No parameter help available
- Power_Dynamics: bool: No parameter help available
- Tx_Power: bool: No parameter help available

get_aclr() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:ACLR
value: bool = driver.configure.nrSubMeas.multiEval.result.get_aclr()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.

Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

return

enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_all() → AllStruct

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

No command help available

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

get_es_flatness() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:ESFLatness
value: bool = driver.configure.nrSubMeas.multiEval.result.get_es_flatness()
```

No command help available

return
enable: No help available

get_evmc() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:EVMC
value: bool = driver.configure.nrSubMeas.multiEval.result.get_evmc()
```

No command help available

return
enable: No help available

get_iemissions() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:IEMissions
value: bool = driver.configure.nrSubMeas.multiEval.result.get_iemissions()
```

No command help available

return
enable: No help available

get_iq() → bool

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

No command help available

return
enable: No help available

get_merror() → bool

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

No command help available

return
enable: No help available

get_modulation() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:MODulation
value: bool = driver.configure.nrSubMeas.multiEval.result.get_modulation()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_pdynamics() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:PDYNamics
value: bool = driver.configure.nrSubMeas.multiEval.result.get_pdynamics()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_perror() → bool

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

No command help available

return
enable: No help available

get_pmonitor() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:PMONitor
value: bool = driver.configure.nrSubMeas.multiEval.result.get_pmonitor()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

`get_se_mask()` → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:SEMask
value: bool = driver.configure.nrSubMeas.multiEval.result.get_se_mask()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

`get_tx_power()` → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:TXPower
value: bool = driver.configure.nrSubMeas.multiEval.result.get_tx_power()
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_txm() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:TXM
value: bool = driver.configure.nrSubMeas.multiEval.result.get_txm()
```

No command help available

return
enable: No help available

set_aclr(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:ACLR
driver.configure.nrSubMeas.multiEval.result.set_aclr(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_all(*value: AllStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult[:ALL]
structure = driver.configure.nrSubMeas.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.Power_Monitor: bool = False
structure.Power_Dynamics: bool = False
structure.Tx_Power: bool = False
driver.configure.nrSubMeas.multiEval.result.set_all(value = structure)
```

No command help available

param value
see the help for AllStruct structure arguments.

set_es_flatness(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:ESFLatness
driver.configure.nrSubMeas.multiEval.result.set_es_flatness(enable = False)
```

No command help available

param enable
No help available

set_evmc(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:EVMC
driver.configure.nrSubMeas.multiEval.result.set_evmc(enable = False)
```

No command help available

param enable
No help available

set_iemissions(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:IEMissions
driver.configure.nrSubMeas.multiEval.result.set_iemissions(enable = False)
```

No command help available

param enable
No help available

set_iq(*enable: bool*) → None

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

No command help available

param enable
No help available

set_merror(*enable: bool*) → None

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

No command help available

param enable
No help available

set_modulation(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:MODulation
driver.configure.nrSubMeas.multiEval.result.set_modulation(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results

- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_pdynamics(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:PDYNamics
driver.configure.nrSubMeas.multiEval.result.set_pdynamics(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_perror(*enable: bool*) → None

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

No command help available

param enable
No help available

set_pmonitor(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:PMONitor
driver.configure.nrSubMeas.multiEval.result.set_pmonitor(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask

- PMONitor / Power monitor
- PDYNamics / Power dynamics

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_se_mask(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:SEMask
driver.configure.nrSubMeas.multiEval.result.set_se_mask(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_tx_power(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:TXPower
driver.configure.nrSubMeas.multiEval.result.set_tx_power(enable = False)
```

Enables or disables the evaluation of results in the multi-evaluation measurement.
Table Header: Mnemonic / Description

- MODulation / Modulation results
- ACLR / Adj. channel leakage power ratio
- TXPower / TX power
- SEMask / Spectrum emission mask
- PMONitor / Power monitor
- PDYNamics / Power dynamics

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_txm(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:TXM
driver.configure.nrSubMeas.multiEval.result.set_txm(enable = False)
```

No command help available

param enable
No help available

Cloning the Group

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

Subgroups

6.1.1.6.13.1 EvMagnitude

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
```

class EvMagnitudeCls

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

get_value() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
value: bool = driver.configure.nrSubMeas.multiEval.result.evMagnitude.get_
↪value()
```

No command help available

return
enable: No help available

set_value(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:RESult:EVMagnitude
driver.configure.nrSubMeas.multiEval.result.evMagnitude.set_value(enable =␣
↪False)
```

No command help available

param enable
No help available

Cloning the Group

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

Subgroups

6.1.1.6.13.2 EvmSymbol

SCPI Command :

```
CONFigure:NRSub: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: No parameter help available
- Symbol: int: No parameter help available
- Low_High: enums.LowHigh: No parameter help available

get() → EvmSymbolStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:RESult:EVMagnitude:EVMSymbol
value: EvmSymbolStruct = driver.configure.nrSubMeas.multiEval.result.
↪evMagnitude.evmSymbol.get()
```

No command help available

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

set(*enable: bool*, *symbol: int*, *low_high: LowHigh*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:RESult:EVMagnitude:EVMSymbol
driver.configure.nrSubMeas.multiEval.result.evMagnitude.evmSymbol.set(enable =␣
↪False, symbol = 1, low_high = enums.LowHigh.HIGH)
```

No command help available

param enable
No help available

param symbol
No help available

param low_high
No help available

6.1.1.6.14 Scount

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:MODulation
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:POWer
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:PDYNamics
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:TXPower
```

class ScountCls

Scount commands group definition. 6 total commands, 1 Subgroups, 4 group commands

get_modulation() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:MODulation
value: int = driver.configure.nrSubMeas.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_pdynamics() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:PDYNamics
value: int = driver.configure.nrSubMeas.multiEval.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

get_power() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:POWer
value: int = driver.configure.nrSubMeas.multiEval.scount.get_power()
```

No command help available

return
statistic_count: No help available

get_tx_power() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:TXPower
value: int = driver.configure.nrSubMeas.multiEval.scount.get_tx_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: Number of measurement intervals

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:MODulation
driver.configure.nrSubMeas.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_pdynamics(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:PDYNamics
driver.configure.nrSubMeas.multiEval.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

set_power(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:POWer
driver.configure.nrSubMeas.multiEval.scount.set_power(statistic_count = 1)
```

No command help available

param statistic_count
No help available

set_tx_power(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:TXPower
driver.configure.nrSubMeas.multiEval.scount.set_tx_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
Number of measurement intervals

Cloning the Group

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

Subgroups

6.1.1.6.14.1 Spectrum

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
value: int = driver.configure.nrSubMeas.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: Number of measurement intervals (slots)

get_se_mask() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
value: int = driver.configure.nrSubMeas.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: Number of measurement intervals (slots)

set_aclr(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:ACLR
driver.configure.nrSubMeas.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
Number of measurement intervals (slots)

set_se_mask(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SCOunt:SPECtrum:SEMask
driver.configure.nrSubMeas.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
Number of measurement intervals (slots)

6.1.1.6.15 Spectrum

class SpectrumCls

Spectrum 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.nrSubMeas.multiEval.spectrum.clone()
```

Subgroups

6.1.1.6.15.1 Aclr

class AclrCls

Aclr commands group definition. 2 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.1.1.6.15.2 Enable

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
CONFigure:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle:ENDC
```

class EnableCls

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

class EnableStruct

Response structure. Fields:

- Utra_1: bool: No parameter help available
- Utra_2: bool: No parameter help available
- Nr: bool: No parameter help available

get() → EnableStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
value: EnableStruct = driver.configure.nrSubMeas.multiEval.spectrum.aclr.enable.
↪get()
```

Enables or disables the evaluation of the first adjacent UTRA channels, second adjacent UTRA channels and first adjacent NR channels.

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

get_endc() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:SPECtrum:ACLR:ENABle:ENDC
value: bool = driver.configure.nrSubMeas.multiEval.spectrum.aclr.enable.get_
↪endc()
```

Enables or disables the evaluation of the adjacent channel power in EN-DC mode.

return
endc: No help available

set(*utra_1: bool*, *utra_2: bool*, *nr: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:ACLR:ENABle
driver.configure.nrSubMeas.multiEval.spectrum.aclr.enable.set(utra_1 = False,␣
↪utra_2 = False, nr = False)
```

Enables or disables the evaluation of the first adjacent UTRA channels, second adjacent UTRA channels and first adjacent NR channels.

param utra_1
No help available

param utra_2
No help available

param nr
No help available

set_endc(*endc: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:SPECtrum:ACLR:ENABle:ENDC
driver.configure.nrSubMeas.multiEval.spectrum.aclr.enable.set_endc(endc = False)
```

Enables or disables the evaluation of the adjacent channel power in EN-DC mode.

param endc
No help available

6.1.1.6.15.3 SeMask

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:SPECtrum:SEMask:MFILter
value: enums.MeasFilter = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>
↪:MEValuation:SPECtrum:SEMask:MFILter
driver.configure.nrSubMeas.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

6.1.1.6.16 Trace

class TraceCls

Trace 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.nrSubMeas.multiEval.trace.clone()
```

Subgroups

6.1.1.6.16.1 Iemissions

class IemissionsCls

Iemissions 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.nrSubMeas.multiEval.trace.iemissions.clone()
```

Subgroups

6.1.1.6.16.2 Vresults

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:TRACe:IEMissions:VRESults
```

class VresultsCls

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

class VresultsStruct

Response structure. Fields:

- Cc_1_Start: int: No parameter help available
- Cc_1_Length: int: No parameter help available
- Cc_2_Start: int: No parameter help available
- Cc_2_Length: int: No parameter help available

get() → VresultsStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:TRACe:IEMissions:VRESults
value: VresultsStruct = driver.configure.nrSubMeas.multiEval.trace.iemissions.
↪vresults.get()
```

No command help available

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

set(*cc_1_start: int*, *cc_1_length: int*, *cc_2_start: int*, *cc_2_length: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:MEValuation:TRACe:IEMissions:VRESults
driver.configure.nrSubMeas.multiEval.trace.iemissions.vresults.set(cc_1_start =␣
↪1, cc_1_length = 1, cc_2_start = 1, cc_2_length = 1)
```

No command help available

param cc_1_start
No help available

param cc_1_length
No help available

param cc_2_start
No help available

param cc_2_length
No help available

6.1.1.7 Network

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:NETWork:FREQuency
CONFigure:NRSub:MEASurement<Instance>:NETWork:BAND
CONFigure:NRSub:MEASurement<Instance>:NETWork:DMODe
CONFigure:NRSub:MEASurement<Instance>:NETWork:NSValue
CONFigure:NRSub:MEASurement<Instance>:NETWork:NANTenna
CONFigure:NRSub:MEASurement<Instance>:NETWork:RFPSharing
```

class NetworkCls

Network commands group definition. 12 total commands, 4 Subgroups, 6 group commands

get_band() → Band

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

No command help available

return
band: No help available

get_dmode() → DuplexModeB

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:DMODe
value: enums.DuplexModeB = driver.configure.nrSubMeas.network.get_dmode()
```

No command help available

return
mode: No help available

get_frequency() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:FREQuency
value: float = driver.configure.nrSubMeas.network.get_frequency()
```

No command help available

return
analyzer_freq: No help available

get_nantenna() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:NANTenna
value: int = driver.configure.nrSubMeas.network.get_nantenna()
```

No command help available

return
number: No help available

get_ns_value() → NetworkSigVal

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:NSValue
value: enums.NetworkSigVal = driver.configure.nrSubMeas.network.get_ns_value()
```

No command help available

return
value: No help available

get_rfp_sharing() → Sharing

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:RFPSharing
value: enums.Sharing = driver.configure.nrSubMeas.network.get_rfp_sharing()
```

No command help available

return
sharing: No help available

set_band(*band: Band*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:BAND
driver.configure.nrSubMeas.network.set_band(band = enums.Band.OB1)
```

No command help available

param band
No help available

set_dmode(*mode: DuplexModeB*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:DMODe
driver.configure.nrSubMeas.network.set_dmode(mode = enums.DuplexModeB.FDD)
```

No command help available

param mode
No help available

set_frequency(*analyzer_freq: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:FREQuency
driver.configure.nrSubMeas.network.set_frequency(analyzer_freq = 1.0)
```

No command help available

param analyzer_freq
No help available

set_nantenna(*number: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:NANTenna
driver.configure.nrSubMeas.network.set_nantenna(number = 1)
```

No command help available

param number
No help available

set_ns_value(*value: NetworkSigVal*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:NSValue
driver.configure.nrSubMeas.network.set_ns_value(value = enums.NetworkSigVal.
↪NS01)
```

No command help available

param value
No help available

set_rfp_sharing(*sharing: Sharing*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:RFPSharing
driver.configure.nrSubMeas.network.set_rfp_sharing(sharing = enums.Sharing.
↪FSHared)
```

No command help available

param sharing
No help available

Cloning the Group

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

Subgroups

6.1.1.7.1 Cc<CarrierComponentOne>

RepCap Settings

```
# Range: Nr1 .. Nr1
rc = driver.configure.nrSubMeas.network.cc.repcap_carrierComponentOne_get()
driver.configure.nrSubMeas.network.cc.repcap_carrierComponentOne_set(repcap.
↪CarrierComponentOne.Nr1)
```

class CcCls

Cc commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: CarrierComponentOne, default value after init: CarrierComponentOne.Nr1

Cloning the Group

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

Subgroups

6.1.1.7.1.1 Cbandwidth

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:NETWork[:CC<no>]:CBANdwidth
```

class CbandwidthCls

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

get(*carrierComponentOne=CarrierComponentOne.Default*) → ChannelBwidth

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork[:CC<no>]:CBANdwidth
value: enums.ChannelBwidth = driver.configure.nrSubMeas.network.cc.cbandwidth.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Default)
```

No command help available

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

return
channel_bw: No help available

set(*channel_bw: ChannelBwidth*, *carrierComponentOne=CarrierComponentOne.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork[:CC<no>]:CBANdwidth
driver.configure.nrSubMeas.network.cc.cbandwidth.set(channel_bw = enums.
↪ChannelBwidth.B005, carrierComponentOne = repcap.CarrierComponentOne.Default)
```

No command help available

param channel_bw
No help available

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

6.1.1.7.1.2 PlcId

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:NETWork[:CC<no>]:PLCid
```

class PlcIdCls

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

get(*carrierComponentOne=CarrierComponentOne.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork[:CC<no>]:PLCid
value: int = driver.configure.nrSubMeas.network.cc.plcId.
↪get(carrierComponentOne = repcap.CarrierComponentOne.Default)
```

No command help available

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

return
physical_cell_id: No help available

set(*physical_cell_id: int*, *carrierComponentOne=CarrierComponentOne.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork[:CC<no>]:PLCid
driver.configure.nrSubMeas.network.cc.plcId.set(physical_cell_id = 1,␣
↪carrierComponentOne = repcap.CarrierComponentOne.Default)
```

No command help available

param physical_cell_id
No help available

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

6.1.1.7.2 Ccall

class CcallCls

Ccall 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.nrSubMeas.network.ccall.clone()
```

Subgroups

6.1.1.7.2.1 TxBwidth

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:NETWork:CCALl:TXBWidth:SCSPacing
```

class TxBwidthCls

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

get_sc_spacing() → SubCarrSpacing

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:CCALl:TXBWidth:SCSPacing
value: enums.SubCarrSpacing = driver.configure.nrSubMeas.network.ccall.txBwidth.
↪get_sc_spacing()
```

No command help available

return
used_scs: No help available

set_sc_spacing(*used_scs: SubCarrSpacing*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:CCALl:TXBWidth:SCSPacing
driver.configure.nrSubMeas.network.ccall.txBwidth.set_sc_spacing(used_scs =␣
↪enums.SubCarrSpacing.S15K)
```

No command help available

param used_scs
No help available

6.1.1.7.3 Prach

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:NETWork:PRACh:PCINdex
```

class PrachCls

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

get_pc_index() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:PRACh:PCINdex
value: int = driver.configure.nrSubMeas.network.prach.get_pc_index()
```

No command help available

return
prach_conf_index: No help available

set_pc_index(*prach_conf_index: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:PRACh:PCINdex
driver.configure.nrSubMeas.network.prach.set_pc_index(prach_conf_index = 1)
```

No command help available

param prach_conf_index
No help available

6.1.1.7.4 UlDl

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:NETWork:ULDL:PERiodicity
```

class UlDlCls

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

get_periodicity() → Periodicity

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:ULDL:PERiodicity
value: enums.Periodicity = driver.configure.nrSubMeas.network.ulDl.get_
↪periodicity()
```

No command help available

return
periodicity: No help available

set_periodicity(*periodicity: Periodicity*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:ULDL:PERiodicity
driver.configure.nrSubMeas.network.ulDl.set_periodicity(periodicity = enums.
↪Periodicity.MS05)
```

No command help available

param periodicity
No help available

Cloning the Group

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

Subgroups

6.1.1.7.4.1 Pattern

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:NETWork:ULDL:PATTern
```

class PatternCls

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

class GetStruct

Response structure. Fields:

- Dl_Slots: int: No parameter help available
- Dl_Symbols: int: No parameter help available

- Ul_Slots: int: No parameter help available
- Ul_Symbols: int: No parameter help available

get(*sc_spacing: SubCarrSpacing*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:ULDL:PATTern
value: GetStruct = driver.configure.nrSubMeas.network.ulDl.pattern.get(sc_
↪spacing = enums.SubCarrSpacing.S15K)
```

No command help available

param sc_spacing
No help available

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

set(*sc_spacing: SubCarrSpacing*, *dl_slots: int*, *dl_symbols: int*, *ul_slots: int*, *ul_symbols: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:NETWork:ULDL:PATTern
driver.configure.nrSubMeas.network.ulDl.pattern.set(sc_spacing = enums.
↪SubCarrSpacing.S15K, dl_slots = 1, dl_symbols = 1, ul_slots = 1, ul_symbols =␣
↪1)
```

No command help available

param sc_spacing
No help available

param dl_slots
No help available

param dl_symbols
No help available

param ul_slots
No help available

param ul_symbols
No help available

6.1.1.8 Prach

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:TOUT
CONFigure:NRSub:MEASurement<Instance>:PRACh:REPetition
CONFigure:NRSub:MEASurement<Instance>:PRACh:SCONdition
CONFigure:NRSub:MEASurement<Instance>:PRACh:MOEXception
CONFigure:NRSub:MEASurement<Instance>:PRACh:PCINdex
CONFigure:NRSub:MEASurement<Instance>:PRACh:PFORmat
CONFigure:NRSub:MEASurement<Instance>:PRACh:SSYMbol
CONFigure:NRSub:MEASurement<Instance>:PRACh:SCSPacing
CONFigure:NRSub:MEASurement<Instance>:PRACh:NOPReambles
CONFigure:NRSub:MEASurement<Instance>:PRACh:POPReambles
CONFigure:NRSub:MEASurement<Instance>:PRACh:LRSindex
CONFigure:NRSub:MEASurement<Instance>:PRACh:ZCZone
```

class PrachCls

Prach commands group definition. 41 total commands, 7 Subgroups, 12 group commands

get_lrs_index() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LRSindex
value: int = driver.configure.nrSubMeas.prach.get_lrs_index()
```

Specifies the logical root sequence index to be used for generation of the preamble sequence.

return
log_root_seq_index: No help available

get_mo_exception() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:MOEXception
value: bool = driver.configure.nrSubMeas.prach.get_mo_exception()
```

Specifies whether measurement results that the CMP180 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:NRSub:MEASurement<Instance>:PRACh:NOPReambles
value: int = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PCINdex
value: int = driver.configure.nrSubMeas.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.) . The range depends on the duplex mode, see 'Preambles in the time domain'.

return
prach_conf_index: No help available

get_pformat() → PreambleFormat

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:PFORmat
value: enums.PreambleFormat = driver.configure.nrSubMeas.prach.get_pformat()
```

Selects the preamble format. The command is only needed for PRACH configuration indices that allow two preamble formats, see tables in 'Preambles in the time domain'.

return
preamble_format: No help available

get_po_preambles() → PeriodPreamble

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:POPReambles
value: enums.PeriodPreamble = driver.configure.nrSubMeas.prach.get_po_
↪preambles()
```

Specifies the periodicity of preambles to be captured for multi-preamble result squares.

return
period_preamble: 10 ms, 20 ms, 5 ms

get_repetition() → Repeat

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:REPetition
value: enums.Repeat = driver.configure.nrSubMeas.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_sc_spacing() → SubCarrSpacingB

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SCSPacing
value: enums.SubCarrSpacingB = driver.configure.nrSubMeas.prach.get_sc_spacing()
```

Specifies the subcarrier spacing used by the UE for the preambles. The allowed subset of values depends on the preamble format, see Table 'Preambles in the frequency domain'.

return
sub_carr_spacing: 1.25 kHz, 5 kHz, 15 kHz, 30 kHz, 60 kHz

get_scondition() → StopCondition

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SCONdition
value: enums.StopCondition = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:SSYMbol
value: int = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TOUT
value: float = driver.configure.nrSubMeas.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_zc_zone() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:ZCZone
value: int = driver.configure.nrSubMeas.prach.get_zc_zone()
```

Specifies the zero correlation zone config, i.e. which NCS value of an NCS set is used for generation of the preamble sequence.

return
zero_corr_zone: No help available

set_lrs_index(*log_root_seq_index: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LRSindex
driver.configure.nrSubMeas.prach.set_lrs_index(log_root_seq_index = 1)
```

Specifies the logical root sequence index to be used for generation of the preamble sequence.

param log_root_seq_index
No help available

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:MOEXception
driver.configure.nrSubMeas.prach.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the CMP180 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:NRSub:MEASurement<Instance>:PRACh:NOPReambles
driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PCINdex
driver.configure.nrSubMeas.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.) . The range depends on the duplex mode, see 'Preambles in the time domain'.

param prach_conf_index
No help available

set_pformat(*preamble_format: PreambleFormat*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:PFORmat
driver.configure.nrSubMeas.prach.set_pformat(preamble_format = enums.
↪PreambleFormat.PF0)
```

Selects the preamble format. The command is only needed for PRACH configuration indices that allow two preamble formats, see tables in 'Preambles in the time domain'.

param preamble_format
No help available

set_po_preambles(*period_preamble: PeriodPreamble*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:POPReambles
driver.configure.nrSubMeas.prach.set_po_preambles(period_preamble = enums.
↪PeriodPreamble.MS05)
```

Specifies the periodicity of preambles to be captured for multi-preamble result squares.

param period_preamble
10 ms, 20 ms, 5 ms

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:REPetition
driver.configure.nrSubMeas.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_sc_spacing(*sub_carr_spacing: SubCarrSpacingB*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SCSPacing
driver.configure.nrSubMeas.prach.set_sc_spacing(sub_carr_spacing = enums.
↪SubCarrSpacingB.S15K)
```

Specifies the subcarrier spacing used by the UE for the preambles. The allowed subset of values depends on the preamble format, see Table 'Preambles in the frequency domain'.

param sub_carr_spacing
1.25 kHz, 5 kHz, 15 kHz, 30 kHz, 60 kHz

set_scondition(*stop_condition: StopCondition*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SCONdition
driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:SSYMbol
driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TOUT
driver.configure.nrSubMeas.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_zc_zone(*zero_corr_zone: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:ZCZone
driver.configure.nrSubMeas.prach.set_zc_zone(zero_corr_zone = 1)
```

Specifies the zero correlation zone config, i.e. which NCS value of an NCS set is used for generation of the preamble sequence.

param zero_corr_zone
No help available

Cloning the Group

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

Subgroups

6.1.1.8.1 Limit

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:FERRor
```

class LimitCls

Limit commands group definition. 9 total commands, 4 Subgroups, 1 group commands

get_freq_error() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:FERRor
value: float or bool = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:LIMit:FERRor
driver.configure.nrSubMeas.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.nrSubMeas.prach.limit.clone()
```

Subgroups

6.1.1.8.1.1 EvMagnitude

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
value: EvMagnitudeStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:LIMit:EVMagnitude
driver.configure.nrSubMeas.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

6.1.1.8.1.2 Merror

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:PRACh:LIMit:MERRor
value: MerrorStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:LIMit:MERRor
driver.configure.nrSubMeas.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

6.1.1.8.1.3 Pdynamics

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:ENABle
CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:OFFPower
```

class PdynamicsCls

Pdynamics commands group definition. 5 total commands, 2 Subgroups, 2 group commands

get_enable() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:ENABle
value: bool = driver.configure.nrSubMeas.prach.limit.pdynamics.get_enable()
```

Enables or disables the limit check for the power dynamics measurement.

return
enable: No help available

get_off_power() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:OFFPower
value: float = driver.configure.nrSubMeas.prach.limit.pdynamics.get_off_power()
```

Defines an upper limit for the OFF power determined with the power dynamics measurement.

return
off_power: Upper limit before adding the test tolerance

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:ENABle
driver.configure.nrSubMeas.prach.limit.pdynamics.set_enable(enable = False)
```

Enables or disables the limit check for the power dynamics measurement.

param enable
No help available

set_off_power(*off_power: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:OFFPower
driver.configure.nrSubMeas.prach.limit.pdynamics.set_off_power(off_power = 1.0)
```

Defines an upper limit for the OFF power determined with the power dynamics measurement.

param off_power
Upper limit before adding the test tolerance

Cloning the Group

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

Subgroups

6.1.1.8.1.4 EonPower<EonPower>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.configure.nrSubMeas.prach.limit.pdynamics.eonPower.repcap_eonPower_get()
driver.configure.nrSubMeas.prach.limit.pdynamics.eonPower.repcap_eonPower_set(repcap.
↪EonPower.Nr1)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:EONPower<e>
```

class EonPowerCls

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

get(*eonPower=EonPower.Default*) → List[float]

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:EONPower<e>
value: List[float] = driver.configure.nrSubMeas.prach.limit.pdynamics.eonPower.
↪get(eonPower = repcap.EonPower.Default)
```

Defines limits for the ON power determined with the power dynamics measurement. There are two limit values per channel bandwidth.

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

return
on_power: Comma-separated list of 15 values, for the channel bandwidths [MHz]: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100. The effective limit is (OnPower - (9 dB + test tolerance)) to (OnPower + (9 dB + test tolerance)).

set(*on_power: List[float]*, *eonPower=EonPower.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:EONPower<e>
driver.configure.nrSubMeas.prach.limit.pdynamics.eonPower.set(on_power = [1.1,␣
↪2.2, 3.3], eonPower = repcap.EonPower.Default)
```

Defines limits for the ON power determined with the power dynamics measurement. There are two limit values per channel bandwidth.

param on_power
Comma-separated list of 15 values, for the channel bandwidths [MHz]: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100. The effective limit is (OnPower - (9 dB + test tolerance)) to (OnPower + (9 dB + test tolerance)).

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

Cloning the Group

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

6.1.1.8.1.5 Ttolerance

class TtoleranceCls

Ttolerance 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.nrSubMeas.prach.limit.pdynamics.ttolerance.clone()
```

Subgroups

6.1.1.8.1.6 Cbgt

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:TTOLerance:CBGT
```

class CbgtCls

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

class CbgtStruct

Response structure. Fields:

- Tt_Power_Less_3_G: float: Tolerance for carrier center frequencies up to 3 GHz.
- Tt_Power_Great_3_G: float: Tolerance for carrier center frequencies 3 GHz.

get() → CbgtStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:PRACh:LIMit:PDYNamics:TTOLerance:CBGT
value: CbgtStruct = driver.configure.nrSubMeas.prach.limit.pdynamics.ttolerance.
↪cbgt.get()
```

Defines test tolerances for power dynamics limits, for channel BW > 40 MHz.

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

set(*tt_power_less_3_g: float*, *tt_power_great_3_g: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:PRACh:LIMit:PDYNamics:TTOLerance:CBGT
driver.configure.nrSubMeas.prach.limit.pdynamics.ttolerance.cbgt.set(tt_power_
↪less_3_g = 1.0, tt_power_great_3_g = 1.0)
```

Defines test tolerances for power dynamics limits, for channel BW > 40 MHz.

param tt_power_less_3_g
Tolerance for carrier center frequencies up to 3 GHz.

param tt_power_great_3_g
Tolerance for carrier center frequencies 3 GHz.

6.1.1.8.1.7 Cblt

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:LIMit:PDYNamics:TTOLerance:CBLT
```

class CbltCls

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

class CbltStruct

Response structure. Fields:

- Tt_Power_Less_3_G: float: Tolerance for carrier center frequencies up to 3 GHz.
- Tt_Power_Great_3_G: float: Tolerance for carrier center frequencies 3 GHz.

get() → CbltStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:PRACh:LIMit:PDYNamics:TTOLerance:CBLT
value: CbltStruct = driver.configure.nrSubMeas.prach.limit.pdynamics.ttolerance.
↪cblt.get()
```

Defines test tolerances for power dynamics limits, for channel BW up to 40 MHz.

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

set(*tt_power_less_3_g: float*, *tt_power_great_3_g: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:PRACh:LIMit:PDYNamics:TTOLerance:CBLT
driver.configure.nrSubMeas.prach.limit.pdynamics.ttolerance.cblt.set(tt_power_
↪less_3_g = 1.0, tt_power_great_3_g = 1.0)
```

Defines test tolerances for power dynamics limits, for channel BW up to 40 MHz.

param tt_power_less_3_g
Tolerance for carrier center frequencies up to 3 GHz.

param tt_power_great_3_g
Tolerance for carrier center frequencies 3 GHz.

6.1.1.8.1.8 Perror

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:PRACh:LIMit:PERRor
value: PerrorStruct = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:LIMit:PERRor
driver.configure.nrSubMeas.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

6.1.1.8.2 Modulation

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:MODulation:EWPosition
```

class ModulationCls

Modulation commands group definition. 3 total commands, 1 Subgroups, 1 group commands

get_ew_position() → LowHigh

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:MODulation:EWPosition
value: enums.LowHigh = driver.configure.nrSubMeas.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

set_ew_position(*evm_window_pos: LowHigh*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:MODulation:EWPosition
driver.configure.nrSubMeas.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

Cloning the Group

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

Subgroups

6.1.1.8.2.1 EwLength

SCPI Command :

```
CONFigure:NRSub: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:NRSub:MEASurement<Instance>:PRACh:MODulation:EWLength
value: List[int] = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:EWLength
driver.configure.nrSubMeas.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.nrSubMeas.prach.modulation.ewLength.clone()
```

Subgroups

6.1.1.8.2.2 Pformat<PFormat>

RepCap Settings

```
# Range: Nr1 .. Nr13
rc = driver.configure.nrSubMeas.prach.modulation.ewLength.pformat.repcap_pFormat_get()
driver.configure.nrSubMeas.prach.modulation.ewLength.pformat.repcap_pFormat_set(repcap.
↪PFormat.Nr1)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat<no>
```

class PformatCls

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

get(*pFormat=PFormat.Default*) → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat
↪<no>
value: int = driver.configure.nrSubMeas.prach.modulation.ewLength.pformat.
↪get(pFormat = repcap.PFormat.Default)
```

No command help available

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

return
evm_window_length: No help available

set(*evm_window_length: int*, *pFormat=PFormat.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:MODulation:EWLength:PFORmat
↪<no>
driver.configure.nrSubMeas.prach.modulation.ewLength.pformat.set(evm_window_
↪length = 1, pFormat = repcap.PFormat.Default)
```

No command help available

param evm_window_length
No help available

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

Cloning the Group

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

6.1.1.8.3 PfOffset

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:PFOFfset:AUTO
CONFigure:NRSub:MEASurement<Instance>:PRACh:PFOFfset
```

class PfOffsetCls

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

get_auto() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:PFOFfset:AUTO
value: bool = driver.configure.nrSubMeas.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_NrFr1Meas.Configure.NrSubMeas.Prach.PfOffset.value.

return
prach_freq_auto: No help available

get_value() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:PFOFfset
value: int = driver.configure.nrSubMeas.prach.pfOffset.get_value()
```

Specifies the PRACH frequency offset. This setting is only relevant if automatic detection is disabled, see method RsCMPX_NrFr1Meas.Configure.NrSubMeas.Prach.PfOffset.auto.

return
prach_freq_offset: No help available

`set_auto`(*prach_freq_auto: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:PFOFfset:AUTO
driver.configure.nrSubMeas.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_NrFr1Meas.Configure.NrSubMeas.Prach.PfOffset.value.

param prach_freq_auto
No help available

`set_value`(*prach_freq_offset: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:PFOFfset
driver.configure.nrSubMeas.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_NrFr1Meas.Configure.NrSubMeas.Prach.PfOffset.auto.

param prach_freq_offset
No help available

6.1.1.8.4 Result

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:MODulation
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PDYNamics
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult[:ALL]
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:EVMagnitude
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:EVPReamble
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:MERRor
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PERRor
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:IQ
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PVPReamble
CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:TXM
```

`class ResultCls`

Result commands group definition. 10 total commands, 0 Subgroups, 10 group commands

`class AllStruct`

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

- Evm: bool: No parameter help available
- Magnitude_Error: bool: No parameter help available
- Phase_Error: bool: No parameter help available
- Iq: bool: No parameter help available
- Power_Dynamics: bool: No parameter help available
- Tx_Measurement: bool: No parameter help available

- Evm_Vs_Preamble: bool: No parameter help available
- Power_Vs_Preamble: bool: No parameter help available

get_all() → AllStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult[:ALL]
value: AllStruct = driver.configure.nrSubMeas.prach.result.get_all()
```

No command help available

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

get_ev_magnitude() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:EVMagnitude
value: bool = driver.configure.nrSubMeas.prach.result.get_ev_magnitude()
```

No command help available

return
enable: No help available

get_ev_preamble() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:EVPReamble
value: bool = driver.configure.nrSubMeas.prach.result.get_ev_preamble()
```

No command help available

return
enable: No help available

get_iq() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:IQ
value: bool = driver.configure.nrSubMeas.prach.result.get_iq()
```

No command help available

return
enable: No help available

get_merror() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:MERRor
value: bool = driver.configure.nrSubMeas.prach.result.get_merror()
```

No command help available

return
enable: No help available

get_modulation() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:MODulation
value: bool = driver.configure.nrSubMeas.prach.result.get_modulation()
```

Enables or disables the evaluation of modulation results in the PRACH measurement.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_pdynamics() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PDYNamics
value: bool = driver.configure.nrSubMeas.prach.result.get_pdynamics()
```

Enables or disables the evaluation of power dynamics results in the PRACH measurement.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_perror() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PERRor
value: bool = driver.configure.nrSubMeas.prach.result.get_perror()
```

No command help available

return
enable: No help available

get_pv_preamble() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PVPReamble
value: bool = driver.configure.nrSubMeas.prach.result.get_pv_preamble()
```

No command help available

return
enable: No help available

get_txm() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:TXM
value: bool = driver.configure.nrSubMeas.prach.result.get_txm()
```

No command help available

return
enable: No help available

set_all(*value: AllStruct*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult[:ALL]
structure = driver.configure.nrSubMeas.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.nrSubMeas.prach.result.set_all(value = structure)
```

No command help available

param value
see the help for AllStruct structure arguments.

set_ev_magnitude(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:EVMagnitude
driver.configure.nrSubMeas.prach.result.set_ev_magnitude(enable = False)
```

No command help available

param enable
No help available

set_ev_preamble(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:EVPReamble
driver.configure.nrSubMeas.prach.result.set_ev_preamble(enable = False)
```

No command help available

param enable
No help available

set_iq(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:IQ
driver.configure.nrSubMeas.prach.result.set_iq(enable = False)
```

No command help available

param enable
No help available

set_merror(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:MERRor
driver.configure.nrSubMeas.prach.result.set_merror(enable = False)
```

No command help available

param enable
No help available

set_modulation(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:MODulation
driver.configure.nrSubMeas.prach.result.set_modulation(enable = False)
```

Enables or disables the evaluation of modulation results in the PRACH measurement.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_pdynamics(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PDYNamics
driver.configure.nrSubMeas.prach.result.set_pdynamics(enable = False)
```

Enables or disables the evaluation of power dynamics results in the PRACH measurement.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_perror(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PERRor
driver.configure.nrSubMeas.prach.result.set_perror(enable = False)
```

No command help available

param enable
No help available

set_pv_preamble(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:PVPReamble
driver.configure.nrSubMeas.prach.result.set_pv_preamble(enable = False)
```

No command help available

param enable
No help available

set_txm(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RESult:TXM
driver.configure.nrSubMeas.prach.result.set_txm(enable = False)
```

No command help available

param enable
No help available

6.1.1.8.5 Rsetting

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:RSETting
```

class RsettingCls

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

get(*restricted_set: RestrictedSet*) → RestrictedSet

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:RSETting
value: enums.RestrictedSet = driver.configure.nrSubMeas.prach.rsetting.
↪get(restricted_set = enums.RestrictedSet.URES)
```

No command help available

param restricted_set
No help available

return
restricted_set: No help available

6.1.1.8.6 Scount

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:SCOunt:MODulation
CONFigure:NRSub:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
```

class ScountCls

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

get_modulation() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SCOunt:MODulation
value: int = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
value: int = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:SCOunt:MODulation
driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:SCOunt:PDYNamics
driver.configure.nrSubMeas.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

6.1.1.8.7 Sindex

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:PRACh:SINDex:AUTO
CONFigure:NRSub:MEASurement<Instance>:PRACh:SINDex
```

class SindexCls

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

get_auto() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SINDex:AUTO
value: bool = driver.configure.nrSubMeas.prach.sindex.get_auto()
```

Enables or disables automatic detection of the sequence index. To configure the index manually for disabled automatic detection, see method RsCMPX_NrFr1Meas.Configure.NrSubMeas.Prach.Sindex.value.

return
seq_index_auto: No help available

get_value() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SINDex
value: int = driver.configure.nrSubMeas.prach.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_NrFr1Meas.Configure.NrSubMeas.Prach.Sindex.auto.

return
sequence_index: No help available

set_auto(*seq_index_auto: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SINDex:AUTO
driver.configure.nrSubMeas.prach.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_NrFr1Meas.Configure.NrSubMeas.Prach.Sindex.value.

param seq_index_auto
No help available

set_value(*sequence_index: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:PRACh:SINDex
driver.configure.nrSubMeas.prach.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_NrFr1Meas.Configure.NrSubMeas.Prach.Sindex.auto.

param sequence_index
No help available

6.1.1.9 RfSettings

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:RFSettings:UMARgin
CONFigure:NRSub:MEASurement<Instance>:RFSettings:ENPower
CONFigure:NRSub:MEASurement<Instance>:RFSettings:FREQuency
CONFigure:NRSub:MEASurement<Instance>:RFSettings:FOFFset
CONFigure:NRSub:MEASurement<Instance>:RFSettings:MLOFfset
CONFigure:NRSub:MEASurement<Instance>:RFSettings:LRINterval
```

class RfSettingsCls

RfSettings commands group definition. 8 total commands, 2 Subgroups, 6 group commands

get_envelope_power() → float

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

Sets the expected nominal power of the measured RF signal.

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:NRSub:MEASurement<Instance>:RFSettings:FOFFset
value: int = driver.configure.nrSubMeas.rfSettings.get_foffset()
```

No command help available

return
offset: No help available

get_frequency() → float

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

No command help available

return
analyzer_freq: No help available

get_lr_interval() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:LRINterval
value: float = driver.configure.nrSubMeas.rfSettings.get_lr_interval()
```

Defines the measurement interval for level adjustment.

return
lvl_rang_interval: No help available

get_ml_offset() → float

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

No command help available

return
mix_lev_offset: No help available

get_umargin() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:UMARgin
value: float = driver.configure.nrSubMeas.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.

return
user_margin: No help available

set_envelope_power(*exp_nom_pow: float*) → None

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

Sets the expected nominal power of the measured RF signal.

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:NRSub:MEASurement<Instance>:RFSettings:FOFFset
driver.configure.nrSubMeas.rfSettings.set_foffset(offset = 1)
```

No command help available

param offset
No help available

set_frequency(*analyzer_freq: float*) → None

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

No command help available

param analyzer_freq
No help available

set_lr_interval(*lvl_rang_interval: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:LRINterval
driver.configure.nrSubMeas.rfSettings.set_lr_interval(lvl_rang_interval = 1.0)
```

Defines the measurement interval for level adjustment.

param lvl_rang_interval
No help available

set_ml_offset(*mix_lev_offset: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:MLOFfset
driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:RFSettings:UMARgin
driver.configure.nrSubMeas.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.

param user_margin
No help available

Cloning the Group

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

Subgroups

6.1.1.9.1 Eattenuation

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:RFSettings:EATTenuation
```

class EattenuationCls

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

class EattenuationStruct

Response structure. Fields:

- Rf_Input_Ext_Att: float: For input path with antenna 1
- Rf_Input_Ext_Att_2: float: For input path with antenna 2

get() → EattenuationStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:EATTenuation
value: EattenuationStruct = driver.configure.nrSubMeas.rfSettings.eattenuation.
↪get()
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector. A query returns 1 or 2 values, depending on the number of antennas configured via method RsCMPX_NrFr1Meas.Configure. NrSubMeas.nantenna.

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

set(*rf_input_ext_att: float*, *rf_input_ext_att_2: float = None*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:EATTenuation
driver.configure.nrSubMeas.rfSettings.eattenuation.set(rf_input_ext_att = 1.0,␣
↪rf_input_ext_att_2 = 1.0)
```

Defines an external attenuation (or gain, if the value is negative) , to be applied to the input connector. A query returns 1 or 2 values, depending on the number of antennas configured via method RsCMPX_NrFr1Meas.Configure. NrSubMeas.nantenna.

param rf_input_ext_att
For input path with antenna 1

param rf_input_ext_att_2
For input path with antenna 2

6.1.1.9.2 LrStart

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:RFSettings:LRSTart
```

class LrStartCls

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

set() → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:LRSTart
driver.configure.nrSubMeas.rfSettings.lrStart.set()
```

Starts level adjustment.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:RFSettings:LRSTart
driver.configure.nrSubMeas.rfSettings.lrStart.set_with_opc()
```

Starts level adjustment.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_NrFr1Meas.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.1.1.10 Srs

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:TOUT
CONFigure:NRSub:MEASurement<Instance>:SRS:REPetition
CONFigure:NRSub:MEASurement<Instance>:SRS:SCONdition
CONFigure:NRSub:MEASurement<Instance>:SRS:MOEXception
CONFigure:NRSub:MEASurement<Instance>:SRS:SEQuence
CONFigure:NRSub:MEASurement<Instance>:SRS:SPOSition
CONFigure:NRSub:MEASurement<Instance>:SRS:NOSYmbols
CONFigure:NRSub:MEASurement<Instance>:SRS:PERiodicity
CONFigure:NRSub:MEASurement<Instance>:SRS:NOSubframes
```

class SrsCls

Srs commands group definition. 30 total commands, 8 Subgroups, 9 group commands

get_mo_exception() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MOEXception
value: bool = driver.configure.nrSubMeas.srs.get_mo_exception()
```

Specifies whether measurement results that the CMP180 identifies as faulty or inaccurate are rejected.

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

get_no_sub_frames() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:NOSubframes
value: int = driver.configure.nrSubMeas.srs.get_no_sub_frames()
```

Configures the number of subframes captured for the power vs symbol result diagram.

return
number_subframes: No help available

get_no_symbols() → NumberSymbols

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:NOSYmbols
value: enums.NumberSymbols = driver.configure.nrSubMeas.srs.get_no_symbols()
```

No command help available

return
number_symbols: No help available

get_periodicity() → SrsPeriodicity

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:PERiodicity
value: enums.SrsPeriodicity = driver.configure.nrSubMeas.srs.get_periodicity()
```

Configures the periodicity of the SRS transmissions in slots.

return
periodicity: No help available

get_repetition() → Repeat

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:REPetition
value: enums.Repeat = driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:SCONdition
value: enums.StopCondition = driver.configure.nrSubMeas.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_sequence() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SEQuence
value: int = driver.configure.nrSubMeas.srs.get_sequence()
```

No command help available

return
sequence_id: No help available

get_sposition() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SPOSition
value: int = driver.configure.nrSubMeas.srs.get_sposition()
```

No command help available

return
start_position: No help available

get_timeout() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:TOUT
value: float = driver.configure.nrSubMeas.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

set_mo_exception(*meas_on_exception: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MOEXception
driver.configure.nrSubMeas.srs.set_mo_exception(meas_on_exception = False)
```

Specifies whether measurement results that the CMP180 identifies as faulty or inaccurate are rejected.

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

set_no_sub_frames(*number_subframes: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:NOSubframes
driver.configure.nrSubMeas.srs.set_no_sub_frames(number_subframes = 1)
```

Configures the number of subframes captured for the power vs symbol result diagram.

param number_subframes
No help available

set_no_symbols(*number_symbols: NumberSymbols*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:NOSYmbols
driver.configure.nrSubMeas.srs.set_no_symbols(number_symbols = enums.
↪NumberSymbols.N1)
```

No command help available

param number_symbols
No help available

set_periodicity(*periodicity: SrsPeriodicity*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:PERiodicity
driver.configure.nrSubMeas.srs.set_periodicity(periodicity = enums.
↪SrsPeriodicity.SL1)
```

Configures the periodicity of the SRS transmissions in slots.

param periodicity
No help available

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:REPetition
driver.configure.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:SCONdition
driver.configure.nrSubMeas.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_sequence(*sequence_id: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SEQuence
driver.configure.nrSubMeas.srs.set_sequence(sequence_id = 1)
```

No command help available

param sequence_id
No help available

set_sposition(*start_position: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SPOSition
driver.configure.nrSubMeas.srs.set_sposition(start_position = 1)
```

No command help available

param start_position
No help available

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:TOUT
driver.configure.nrSubMeas.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

Cloning the Group

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

Subgroups

6.1.1.10.1 Limit

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:FERRor
```

class LimitCls

Limit commands group definition. 9 total commands, 4 Subgroups, 1 group commands

get_freq_error() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:FERRor
value: float or bool = driver.configure.nrSubMeas.srs.limit.get_freq_error()
```

No command help available

return
frequency_error: (float or boolean) No help available

set_freq_error(*frequency_error: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:FERRor
driver.configure.nrSubMeas.srs.limit.set_freq_error(frequency_error = 1.0)
```

No command help available

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.nrSubMeas.srs.limit.clone()
```

Subgroups

6.1.1.10.1.1 EvMagnitude

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS: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:NRSub:MEASurement<Instance>:SRS:LIMit:EVMagnitude
value: EvMagnitudeStruct = driver.configure.nrSubMeas.srs.limit.evMagnitude.
↪get()
```

No command help available

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:EVMagnitude
driver.configure.nrSubMeas.srs.limit.evMagnitude.set(rms = 1.0, peak = 1.0)
```

No command help available

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.10.1.2 Merror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS: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:NRSub:MEASurement<Instance>:SRS:LIMit:MERRor
value: MerrorStruct = driver.configure.nrSubMeas.srs.limit.merror.get()
```

No command help available

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:MERRor
driver.configure.nrSubMeas.srs.limit.merror.set(rms = 1.0, peak = 1.0)
```

No command help available

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.10.1.3 Pdynamics

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:ENABle
CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:OFFPower
```

class PdynamicsCls

Pdynamics commands group definition. 5 total commands, 2 Subgroups, 2 group commands

get_enable() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:ENABle
value: bool = driver.configure.nrSubMeas.srs.limit.pdynamics.get_enable()
```

Enables or disables the limit check for the power dynamics measurement.

return
enable: No help available

get_off_power() → float

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:OFFPower
value: float = driver.configure.nrSubMeas.srs.limit.pdynamics.get_off_power()
```

Defines an upper limit for the OFF power determined with the power dynamics measurement.

return
off_power: Upper limit before adding the test tolerance

set_enable(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:ENABle
driver.configure.nrSubMeas.srs.limit.pdynamics.set_enable(enable = False)
```

Enables or disables the limit check for the power dynamics measurement.

param enable
No help available

set_off_power(*off_power: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:OFFPower
driver.configure.nrSubMeas.srs.limit.pdynamics.set_off_power(off_power = 1.0)
```

Defines an upper limit for the OFF power determined with the power dynamics measurement.

param off_power
Upper limit before adding the test tolerance

Cloning the Group

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

Subgroups

6.1.1.10.1.4 EonPower<EonPowerScs>

RepCap Settings

```
# Range: Nr15 .. Nr60
rc = driver.configure.nrSubMeas.srs.limit.pdynamics.eonPower.repcap_eonPowerScs_get()
driver.configure.nrSubMeas.srs.limit.pdynamics.eonPower.repcap_eonPowerScs_set(repcap.
↪EonPowerScs.Nr15)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:EONPower<scs>
```

class EonPowerCls

EonPower commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: EonPowerScs, default value after init: EonPowerScs.Nr15

get(*eonPowerScs=EonPowerScs.Default*) → List[float]

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:EONPower<scs>
value: List[float] = driver.configure.nrSubMeas.srs.limit.pdynamics.eonPower.
↪get(eonPowerScs = repcap.EonPowerScs.Default)
```

Defines limits for the ON power determined with the power dynamics measurement. There is one limit value per SCS and channel bandwidth. The effective limit is (<OnPower> - (9 dB + test tolerance)) to (<OnPower> + (9 dB + test tolerance)).

param eonPowerScs
optional repeated capability selector. Default value: Nr15 (settable in the interface 'EonPower')

return
on_power: Comma-separated list of 15 values, for the channel bandwidths [MHz]: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100. For 15 kHz, the highest five

channel bandwidths are not defined. For 60 kHz, the lowest channel bandwidth is not defined. Send KEEP for not defined values.

set(*on_power: List[float]*, *eonPowerScs=EonPowerScs.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:EONPower<scs>
driver.configure.nrSubMeas.srs.limit.pdynamics.eonPower.set(on_power = [1.1, 2.
↪2, 3.3], eonPowerScs = repcap.EonPowerScs.Default)
```

Defines limits for the ON power determined with the power dynamics measurement. There is one limit value per SCS and channel bandwidth. The effective limit is (<OnPower> - (9 dB + test tolerance)) to (<OnPower> + (9 dB + test tolerance)).

param on_power
Comma-separated list of 15 values, for the channel bandwidths [MHz]: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100. For 15 kHz, the highest five channel bandwidths are not defined. For 60 kHz, the lowest channel bandwidth is not defined. Send KEEP for not defined values.

param eonPowerScs
optional repeated capability selector. Default value: Nr15 (settable in the interface 'EonPower')

Cloning the Group

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

6.1.1.10.1.5 Ttolerance

class TtoleranceCls

Ttolerance 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.nrSubMeas.srs.limit.pdynamics.ttolerance.clone()
```

Subgroups

6.1.1.10.1.6 Cbgt

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:TTOLerance:CBGT
```

class CbgtCls

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

class CbgtStruct

Response structure. Fields:

- Tt_Power_Less_3_G: float: Tolerance for carrier center frequencies up to 3 GHz.
- Tt_Power_Great_3_G: float: Tolerance for carrier center frequencies 3 GHz.

get() → CbgtStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:SRS:LIMit:PDYNamics:TTOLerance:CBGT
value: CbgtStruct = driver.configure.nrSubMeas.srs.limit.pdynamics.ttolerance.
↪cbgt.get()
```

Defines test tolerances for power dynamics limits, for channel BW > 40 MHz.

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

set(*tt_power_less_3_g: float*, *tt_power_great_3_g: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:SRS:LIMit:PDYNamics:TTOLerance:CBGT
driver.configure.nrSubMeas.srs.limit.pdynamics.ttolerance.cbgt.set(tt_power_
↪less_3_g = 1.0, tt_power_great_3_g = 1.0)
```

Defines test tolerances for power dynamics limits, for channel BW > 40 MHz.

param tt_power_less_3_g
Tolerance for carrier center frequencies up to 3 GHz.

param tt_power_great_3_g
Tolerance for carrier center frequencies 3 GHz.

6.1.1.10.1.7 Cblt

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PDYNamics:TTOLerance:CBLT
```

class CbltCls

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

class CbltStruct

Response structure. Fields:

- Tt_Power_Less_3_G: float: Tolerance for carrier center frequencies up to 3 GHz.
- Tt_Power_Great_3_G: float: Tolerance for carrier center frequencies 3 GHz.

get() → CbltStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:SRS:LIMit:PDYNamics:TTOLerance:CBLT
value: CbltStruct = driver.configure.nrSubMeas.srs.limit.pdynamics.ttolerance.
↪cblt.get()
```

Defines test tolerances for power dynamics limits, for channel BW up to 40 MHz.

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

set(*tt_power_less_3_g: float*, *tt_power_great_3_g: float*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>
↪:SRS:LIMit:PDYNamics:TTOLerance:CBLT
driver.configure.nrSubMeas.srs.limit.pdynamics.ttolerance.cblt.set(tt_power_
↪less_3_g = 1.0, tt_power_great_3_g = 1.0)
```

Defines test tolerances for power dynamics limits, for channel BW up to 40 MHz.

param tt_power_less_3_g
Tolerance for carrier center frequencies up to 3 GHz.

param tt_power_great_3_g
Tolerance for carrier center frequencies 3 GHz.

6.1.1.10.1.8 Perror

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS: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:NRSub:MEASurement<Instance>:SRS:LIMit:PERRor
value: PerrorStruct = driver.configure.nrSubMeas.srs.limit.perror.get()
```

No command help available

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

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:LIMit:PERRor
driver.configure.nrSubMeas.srs.limit.perror.set(rms = 1.0, peak = 1.0)
```

No command help available

param rms
(float or boolean) No help available

param peak
(float or boolean) No help available

6.1.1.10.2 Modulation

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:SSYMbol
CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:EWPosition
```

class ModulationCls

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

get_ew_position() → LowHigh

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:EWPosition
value: enums.LowHigh = driver.configure.nrSubMeas.srs.modulation.get_ew_
↪position()
```

No command help available

return
evm_window_pos: No help available

get_ssymbol() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:SSYMbol
value: int = driver.configure.nrSubMeas.srs.modulation.get_ssymbol()
```

No command help available

return
selected_symbol: No help available

set_ew_position(*evm_window_pos: LowHigh*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:EWPosition
driver.configure.nrSubMeas.srs.modulation.set_ew_position(evm_window_pos =␣
↪enums.LowHigh.HIGH)
```

No command help available

param evm_window_pos
No help available

set_ssymbol(*selected_symbol: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:SSYMbol
driver.configure.nrSubMeas.srs.modulation.set_ssymbol(selected_symbol = 1)
```

No command help available

param selected_symbol
No help available

Cloning the Group

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

Subgroups

6.1.1.10.2.1 EwLength

class EwLengthCls

EwLength 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.nrSubMeas.srs.modulation.ewLength.clone()
```

Subgroups

6.1.1.10.2.2 Cbandwidth<ChannelBw>

RepCap Settings

```
# Range: Bw5 .. Bw100
rc = driver.configure.nrSubMeas.srs.modulation.ewLength.cbandwidth.repcap_channelBw_get()
driver.configure.nrSubMeas.srs.modulation.ewLength.cbandwidth.repcap_channelBw_
↪set(repcap.ChannelBw.Bw5)
```

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:EWLength:CBANdwidth<bw>
```

class CbandwidthCls

Cbandwidth commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: ChannelBw, default value after init: ChannelBw.Bw5

class CbandwidthStruct

Response structure. Fields:

- Cyc_Prefix_Norm_15: int: No parameter help available
- Cyc_Prefix_Norm_30: int: No parameter help available
- Cyc_Prefix_Norm_60: int: No parameter help available
- Cyc_Prefix_Extend: int: No parameter help available

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

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:EWLength:CBANdwidth
↪<bw>
value: CbandwidthStruct = driver.configure.nrSubMeas.srs.modulation.ewLength.
↪cbandwidth.get(channelBw = repcap.ChannelBw.Default)
```

No command help available

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

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

set(*cyc_prefix_norm_15: int*, *cyc_prefix_norm_30: int*, *cyc_prefix_norm_60: int*, *cyc_prefix_extend: int*, *channelBw=ChannelBw.Default*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:MODulation:EWLength:CBANdwidth
↪<bw>
driver.configure.nrSubMeas.srs.modulation.ewLength.cbandwidth.set(cyc_prefix_
↪norm_15 = 1, cyc_prefix_norm_30 = 1, cyc_prefix_norm_60 = 1, cyc_prefix_
↪extend = 1, channelBw = repcap.ChannelBw.Default)
```

No command help available

param cyc_prefix_norm_15
No help available

param cyc_prefix_norm_30
No help available

param cyc_prefix_norm_60
No help available

param cyc_prefix_extend
No help available

param channelBw
optional repeated capability selector. Default value: Bw5 (settable in the interface 'Cbandwidth')

Cloning the Group

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

6.1.1.10.3 Pdynamics

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:PDYNamics:HDMode
```

class PdynamicsCls

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

get_hdmode() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:PDYNamics:HDMode
value: bool = driver.configure.nrSubMeas.srs.pdynamics.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:HDMode
driver.configure.nrSubMeas.srs.pdynamics.set_hdmode(high_dynamic_mode = False)
```

Enables or disables the high dynamic mode for power dynamics measurements.

param high_dynamic_mode
No help available

6.1.1.10.4 Result

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:MODulation
CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:PDYNamics
CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:PVSYmbol
```

class ResultCls

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

get_modulation() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:MODulation
value: bool = driver.configure.nrSubMeas.srs.result.get_modulation()
```

No command help available

return
enable: No help available

get_pdynamics() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:PDYNamics
value: bool = driver.configure.nrSubMeas.srs.result.get_pdynamics()
```

Enables or disables the evaluation of power dynamics results in the SRS measurement.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

get_pv_symbol() → bool

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:PVSYmbol
value: bool = driver.configure.nrSubMeas.srs.result.get_pv_symbol()
```

Enables or disables the evaluation of power vs symbol results in the SRS measurement.

return
enable: OFF: Do not evaluate the results. ON: Evaluate the results.

set_modulation(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:MODulation
driver.configure.nrSubMeas.srs.result.set_modulation(enable = False)
```

No command help available

param enable
No help available

set_pdynamics(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:PDYNamics
driver.configure.nrSubMeas.srs.result.set_pdynamics(enable = False)
```

Enables or disables the evaluation of power dynamics results in the SRS measurement.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

set_pv_symbol(*enable: bool*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RESult:PVSYmbol
driver.configure.nrSubMeas.srs.result.set_pv_symbol(enable = False)
```

Enables or disables the evaluation of power vs symbol results in the SRS measurement.

param enable
OFF: Do not evaluate the results. ON: Evaluate the results.

6.1.1.10.5 Rmapping

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:RMAPping
```

class RmappingCls

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

class RmappingStruct

Response structure. Fields:

- Number_Symbols: enums.NumberSymbols: Number of OFDM symbols per SRS transmission.

- Start_Position: int: Start position of the SRS resource (0 = last symbol of slot) .

get() → RmappingStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RMAPping
value: RmappingStruct = driver.configure.nrSubMeas.srs.rmapping.get()
```

Defines the position of the SRS resource within a slot.

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

set(*number_symbols: NumberSymbols*, *start_position: int = None*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RMAPping
driver.configure.nrSubMeas.srs.rmapping.set(number_symbols = enums.
↪NumberSymbols.N1, start_position = 1)
```

Defines the position of the SRS resource within a slot.

param number_symbols
Number of OFDM symbols per SRS transmission.

param start_position
Start position of the SRS resource (0 = last symbol of slot) .

6.1.1.10.6 Rtype

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:RTYPe
```

class RtypeCls

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

class RtypeStruct

Response structure. Fields:

- Periodicity: enums.SrsPeriodicity: No parameter help available
- Offset: int: No parameter help available

get() → RtypeStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RTYPe
value: RtypeStruct = driver.configure.nrSubMeas.srs.rtype.get()
```

No command help available

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

set(*periodicity: SrsPeriodicity*, *offset: int = None*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:RTYPe
driver.configure.nrSubMeas.srs.rtype.set(periodicity = enums.SrsPeriodicity.SL1,
↪ offset = 1)
```

No command help available

param periodicity
No help available

param offset
No help available

6.1.1.10.7 Scount

SCPI Commands :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:SCOunt:MODulation
CONFigure:NRSub:MEASurement<Instance>:SRS:SCOunt:PDYNamics
```

class ScountCls

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

get_modulation() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SCOunt:MODulation
value: int = driver.configure.nrSubMeas.srs.scount.get_modulation()
```

No command help available

return
statistic_count: No help available

get_pdynamics() → int

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SCOunt:PDYNamics
value: int = driver.configure.nrSubMeas.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: No help available

set_modulation(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SCOunt:MODulation
driver.configure.nrSubMeas.srs.scount.set_modulation(statistic_count = 1)
```

No command help available

param statistic_count
No help available

set_pdynamics(*statistic_count: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:SCOunt:PDYNamics
driver.configure.nrSubMeas.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
No help available

6.1.1.10.8 Tcomb

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:SRS:TCOMb
```

class TcombCls

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

class TcombStruct

Response structure. Fields:

- Ktc: enums.Ktc: No parameter help available
- Offset: int: No parameter help available
- Cyclic_Shift: int: No parameter help available

get() → TcombStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:TCOMb
value: TcombStruct = driver.configure.nrSubMeas.srs.tcomb.get()
```

No command help available

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

set(*ktc: Ktc*, *offset: int = None*, *cyclic_shift: int = None*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:SRS:TCOMb
driver.configure.nrSubMeas.srs.tcomb.set(ktc = enums.Ktc.N2, offset = 1, cyclic_
↪shift = 1)
```

No command help available

param ktc
No help available

param offset
No help available

param cyclic_shift
No help available

6.1.1.11 UlDl

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:ULDL:PERiodicity
```

class UlDlCls

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

get_periodicity() → Periodicity

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:ULDL:PERiodicity
value: enums.Periodicity = driver.configure.nrSubMeas.ulDl.get_periodicity()
```

Configures the periodicity of the TDD UL-DL pattern.

return
periodicity: 0.5 ms, 1 ms, 1.25 ms, 2 ms, 2.5 ms, 3 ms, 4 ms, 5 ms, 10 ms

set_periodicity(*periodicity: Periodicity*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:ULDL:PERiodicity
driver.configure.nrSubMeas.ulDl.set_periodicity(periodicity = enums.Periodicity.
↪MS05)
```

Configures the periodicity of the TDD UL-DL pattern.

param periodicity
0.5 ms, 1 ms, 1.25 ms, 2 ms, 2.5 ms, 3 ms, 4 ms, 5 ms, 10 ms

Cloning the Group

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

Subgroups

6.1.1.11.1 Pattern

SCPI Command :

```
CONFigure:NRSub:MEASurement<Instance>:ULDL:PATTern
```

class PatternCls

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

class GetStruct

Response structure. Fields:

- Dl_Slots: int: Specifies 'nrofDownlinkSlots'.
- Dl_Symbols: int: Specifies 'nrofDownlinkSymbols'.
- Ul_Slots: int: Specifies 'nrofUplinkSlots'.

- Ul_Symbols: int: Specifies 'nrofUplinkSymbols'.

get(*sc_spacing: SubCarrSpacing*) → GetStruct

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:ULDL:PATTern
value: GetStruct = driver.configure.nrSubMeas.ulDl.pattern.get(sc_spacing =␣
↪enums.SubCarrSpacing.S15K)
```

Configures the TDD UL-DL pattern for the <SCSpacing>. The ranges have dependencies, see 'TDD UL-DL configuration'.

param sc_spacing
Subcarrier spacing for which the other settings apply.

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

set(*sc_spacing: SubCarrSpacing*, *dl_slots: int*, *dl_symbols: int*, *ul_slots: int*, *ul_symbols: int*) → None

```
# SCPI: CONFigure:NRSub:MEASurement<Instance>:ULDL:PATTern
driver.configure.nrSubMeas.ulDl.pattern.set(sc_spacing = enums.SubCarrSpacing.
↪S15K, dl_slots = 1, dl_symbols = 1, ul_slots = 1, ul_symbols = 1)
```

Configures the TDD UL-DL pattern for the <SCSpacing>. The ranges have dependencies, see 'TDD UL-DL configuration'.

param sc_spacing
Subcarrier spacing for which the other settings apply.

param dl_slots
Specifies 'nrofDownlinkSlots'.

param dl_symbols
Specifies 'nrofDownlinkSymbols'.

param ul_slots
Specifies 'nrofUplinkSlots'.

param ul_symbols
Specifies 'nrofUplinkSymbols'.

6.2 NrSubMeas

class NrSubMeasCls

NrSubMeas commands group definition. 722 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1 MultiEval

SCPI Commands :

```
INITiate:NRSub:MEASurement<Instance>:MEValuation
STOP:NRSub:MEASurement<Instance>:MEValuation
ABORt:NRSub:MEASurement<Instance>:MEValuation
```

class MultiEvalCls

MultiEval commands group definition. 566 total commands, 13 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:NRSub:MEASurement<Instance>:MEValuation
driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation
driver.nrSubMeas.multiEval.initiate()


    INTRO_CMD_HELP: Starts, stops or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the RUN state.
    - STOP... halts the measurement immediately. The measurement enters the RDY␣
↪state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the␣
↪OFF state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

stop() → None

```
# SCPI: STOP:NRSub:MEASurement<Instance>:MEValuation
driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation
driver.nrSubMeas.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_NrFr1Meas.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.nrSubMeas.multiEval.clone()
```

Subgroups

6.2.1.1 Aclr

class AclrCls

Aclr commands group definition. 14 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.1.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:ACLR:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:ACLR:AVERage
CALCulate:NRSub: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
- Nr_Neg: enums.ResultStatus2: ACLR for the first NR channel with lower frequency
- Carrier: enums.ResultStatus2: Power in the allocated NR channel
- Nr_Pos: enums.ResultStatus2: ACLR for the first NR 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
- Nr_Neg: float: ACLR for the first NR channel with lower frequency
- Carrier: float: Power in the allocated NR channel
- Nr_Pos: float: ACLR for the first NR 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.aclr.average.calculate()
```

Returns the relative ACLR values for NR standalone, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: ResultData = driver.nrSubMeas.multiEval.aclr.average.fetch()
```

Returns the relative ACLR values for NR standalone, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:AVERage
value: ResultData = driver.nrSubMeas.multiEval.aclr.average.read()
```

Returns the relative ACLR values for NR standalone, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:ACLR:CURRent
CALCulate:NRSub: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
- Nr_Neg: enums.ResultStatus2: ACLR for the first NR channel with lower frequency
- Carrier: enums.ResultStatus2: Power in the allocated NR channel
- Nr_Pos: enums.ResultStatus2: ACLR for the first NR 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
- Nr_Neg: float: ACLR for the first NR channel with lower frequency
- Carrier: float: Power in the allocated NR channel
- Nr_Pos: float: ACLR for the first NR 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.aclr.current.calculate()
```

Returns the relative ACLR values for NR standalone, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: ResultData = driver.nrSubMeas.multiEval.aclr.current.fetch()
```

Returns the relative ACLR values for NR standalone, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:CURRent
value: ResultData = driver.nrSubMeas.multiEval.aclr.current.read()
```

Returns the relative ACLR values for NR standalone, 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:NRSub: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 transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:ACLR:DALLocation
value: FetchStruct = driver.nrSubMeas.multiEval.aclr.dallocation.fetch()
```

Returns the detected allocation for the measured slot.

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

6.2.1.1.4 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:ACLR:DCHType
```

class DchTypeCls

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

fetch() → ChannelTypeA

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:ACLR:DCHType
value: enums.ChannelTypeA = driver.nrSubMeas.multiEval.aclr.dchType.fetch()
```

No command help available

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.1.5 Endc

class EndcCls

Endc 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.nrSubMeas.multiEval.aclr.endc.clone()
```

Subgroups

6.2.1.1.5.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:AVERage
```

class AverageCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Endc_Neg: enums.ResultStatus2: ACLR for the adjacent channel with lower frequency
- Carrier: enums.ResultStatus2: Power in the allocated channel (aggregated BW)

- Endc_Pos: enums.ResultStatus2: ACLR for the adjacent channel with higher frequency

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Endc_Neg: float: ACLR for the adjacent channel with lower frequency
- Carrier: float: Power in the allocated channel (aggregated BW)
- Endc_Pos: float: ACLR for the adjacent channel with higher frequency

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.aclr.endc.average.
↪calculate()
```

Returns the relative ACLR values for EN-DC, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:AVERage
value: ResultData = driver.nrSubMeas.multiEval.aclr.endc.average.fetch()
```

Returns the relative ACLR values for EN-DC, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:AVERage
value: ResultData = driver.nrSubMeas.multiEval.aclr.endc.average.read()
```

Returns the relative ACLR values for EN-DC, 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.

6.2.1.1.5.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:CURRent
```

class CurrentCls

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

class CalculateStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Endc_Neg: enums.ResultStatus2: ACLR for the adjacent channel with lower frequency
- Carrier: enums.ResultStatus2: Power in the allocated channel (aggregated BW)
- Endc_Pos: enums.ResultStatus2: ACLR for the adjacent channel with higher frequency

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Endc_Neg: float: ACLR for the adjacent channel with lower frequency
- Carrier: float: Power in the allocated channel (aggregated BW)
- Endc_Pos: float: ACLR for the adjacent channel with higher frequency

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.aclr.endc.current.
↪calculate()
```

Returns the relative ACLR values for EN-DC, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:CURRent
value: ResultData = driver.nrSubMeas.multiEval.aclr.endc.current.fetch()
```

Returns the relative ACLR values for EN-DC, 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:NRSub:MEASurement<Instance>:MEValuation:ACLR:ENDC:CURRent
value: ResultData = driver.nrSubMeas.multiEval.aclr.endc.current.read()
```

Returns the relative ACLR values for EN-DC, 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.

6.2.1.2 Amarker<AbsoluteMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.amarker.repcap_absoluteMarker_get()
driver.nrSubMeas.multiEval.amarker.repcap_absoluteMarker_set(repcap.AbsoluteMarker.Nr1)
```

class AmarkerCls

Amarker commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: AbsoluteMarker, default value after init: AbsoluteMarker.Nr1

Cloning the Group

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

Subgroups

6.2.1.2.1 Pdynamics

SCPI Command :

```
FETCh:NRSub: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*, *absoluteMarker=AbsoluteMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:AMARker<No>:PDYNamics
value: float = driver.nrSubMeas.multiEval.amarker.pdynamics.fetch(xvalue = 1.0,␣
↪trace_select = enums.TraceSelect.AVERage, absoluteMarker = repcap.
↪AbsoluteMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the power dynamics trace.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position

param trace_select
No help available

param absoluteMarker
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 Pmonitor

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:AMARker<No>:PMONitor
```

class PmonitorCls

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

fetch(*xvalue: int*, *absoluteMarker=AbsoluteMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:AMARker<No>:PMONitor
value: float = driver.nrSubMeas.multiEval.amarker.pmonitor.fetch(xvalue = 1,␣
↪absoluteMarker = repcap.AbsoluteMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the power monitor trace.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position (slot number)

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

return
yvalue: Absolute y-value of the marker position

6.2.1.3 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.cc.repcap_carrierComponent_get()
driver.nrSubMeas.multiEval.cc.repcap_carrierComponent_set(repcap.CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 78 total commands, 2 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.nrSubMeas.multiEval.cc.clone()
```

Subgroups

6.2.1.3.1 Layer<Layer>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.cc.layer.repcap_layer_get()
driver.nrSubMeas.multiEval.cc.layer.repcap_layer_set(repcap.Layer.Nr1)
```

class LayerCls

Layer commands group definition. 75 total commands, 9 Subgroups, 0 group commands Repeated Capability: Layer, default value after init: Layer.Nr1

Cloning the Group

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

Subgroups

6.2.1.3.1.1 Amarker<AbsoluteMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.cc.layer.amarker.repcap_absoluteMarker_get()
driver.nrSubMeas.multiEval.cc.layer.amarker.repcap_absoluteMarker_set(repcap.
↪AbsoluteMarker.Nr1)
```

class AmarkerCls

Amarker commands group definition. 4 total commands, 3 Subgroups, 0 group commands Repeated Capability: AbsoluteMarker, default value after init: AbsoluteMarker.Nr1

Cloning the Group

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

Subgroups

6.2.1.3.1.2 EvMagnitude

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:AMARker<Nr>
↪:EVMagnitude
```

class EvMagnitudeCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *absoluteMarker=AbsoluteMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:AMARker<Nr>:EVMagnitude
value: float = driver.nrSubMeas.multiEval.cc.layer.amarker.evMagnitude.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage, carrierComponent␣
↪= repcap.CarrierComponent.Default, layer = repcap.Layer.Default,␣
↪absoluteMarker = repcap.AbsoluteMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

param absoluteMarker
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.nrSubMeas.multiEval.cc.layer.amarker.evMagnitude.clone()
```

Subgroups

6.2.1.3.1.3 Peak

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:AMARker<Nr>
↪:EVMagnitude:PEAK
```

class PeakCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *absoluteMarker=AbsoluteMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:AMARker<Nr>:EVMagnitude:PEAK
value: float = driver.nrSubMeas.multiEval.cc.layer.amarker.evMagnitude.peak.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage, carrierComponent␣
↪= repcap.CarrierComponent.Default, layer = repcap.Layer.Default,␣
↪absoluteMarker = repcap.AbsoluteMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

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

return
yvalue: Absolute y-value of the marker position

6.2.1.3.1.4 Merror

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:AMARker<Nr>:MERRor
```

class MerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *absoluteMarker=AbsoluteMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:AMARker<Nr>:MERRor
value: float = driver.nrSubMeas.multiEval.cc.layer.amarker.merror.fetch(xvalue␣
↪= 1, trace_select = enums.TraceSelect.AVERage, carrierComponent = repcap.
↪CarrierComponent.Default, layer = repcap.Layer.Default, absoluteMarker =␣
↪repcap.AbsoluteMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

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

return
yvalue: Absolute y-value of the marker position

6.2.1.3.1.5 Perror

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:AMARker<Nr>:PERRor
```

class PerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *absoluteMarker=AbsoluteMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:AMARker<Nr>:PERRor
value: float = driver.nrSubMeas.multiEval.cc.layer.amarker.perror.fetch(xvalue␣
↪= 1, trace_select = enums.TraceSelect.AVERage, carrierComponent = repcap.
↪CarrierComponent.Default, layer = repcap.Layer.Default, absoluteMarker =␣
↪repcap.AbsoluteMarker.Default)
```

Uses the markers 1 and 2 with absolute values on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

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

return
yvalue: Absolute y-value of the marker position

6.2.1.3.1.6 Dmarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.cc.layer.dmarker.repcap_deltaMarker_get()
driver.nrSubMeas.multiEval.cc.layer.dmarker.repcap_deltaMarker_set(repcap.DeltaMarker.
↪Nr1)
```

class DmarkerCls

Dmarker commands group definition. 4 total commands, 3 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.nrSubMeas.multiEval.cc.layer.dmarker.clone()
```

Subgroups

6.2.1.3.1.7 EvMagnitude

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:DMARker<Nr>
↪:EVMagnitude
```

class EvMagnitudeCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:DMARker<Nr>:EVMagnitude
value: float = driver.nrSubMeas.multiEval.cc.layer.dmarker.evMagnitude.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage, carrierComponent␣
↪= repcap.CarrierComponent.Default, layer = repcap.Layer.Default, 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 OFDM symbol.

Suppressed linked return values: reliability

param xvalue
X-value of the marker position relative to the x-value of the reference marker There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

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.nrSubMeas.multiEval.cc.layer.dmarker.evMagnitude.clone()
```

Subgroups

6.2.1.3.1.8 Peak

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:DMARker<Nr>
↪:EVMagnitude:PEAK
```

class PeakCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:DMARker<Nr>:EVMagnitude:PEAK
value: float = driver.nrSubMeas.multiEval.cc.layer.dmarker.evMagnitude.peak.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage, carrierComponent␣
↪= repcap.CarrierComponent.Default, layer = repcap.Layer.Default, 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 OFDM symbol.

Suppressed linked return values: reliability

param xvalue
X-value of the marker position relative to the x-value of the reference marker There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

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

6.2.1.3.1.9 Merror

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:DMARker<Nr>:MERRor
```

class MerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:DMARker<Nr>:MERRor
value: float = driver.nrSubMeas.multiEval.cc.layer.dmarker.merror.fetch(xvalue␣
↪= 1, trace_select = enums.TraceSelect.AVERage, carrierComponent = repcap.
↪CarrierComponent.Default, layer = repcap.Layer.Default, 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 OFDM symbol.

Suppressed linked return values: reliability

param xvalue
X-value of the marker position relative to the x-value of the reference marker There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

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

6.2.1.3.1.10 Perror

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:DMARker<Nr>:PERRor
```

class PerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:DMARker<Nr>:PERRor
value: float = driver.nrSubMeas.multiEval.cc.layer.dmarker.perror.fetch(xvalue␣
↪= 1, trace_select = enums.TraceSelect.AVERage, carrierComponent = repcap.
↪CarrierComponent.Default, layer = repcap.Layer.Default, 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 OFDM symbol.

Suppressed linked return values: reliability

param xvalue
X-value of the marker position relative to the x-value of the reference marker There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

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

6.2.1.3.1.11 EsFlatness

class EsFlatnessCls

EsFlatness 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.nrSubMeas.multiEval.cc.layer.esFlatness.clone()
```

Subgroups

6.2.1.3.1.12 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:ESFLatness:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:ESFLatness:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.esFlatness.average.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return current, average and extreme limit check results for the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.average.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

6.2.1.3.1.13 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:ESFLatness:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:ESFLatness:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.esFlatness.current.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return current, average and extreme limit check results for the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.current.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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.nrSubMeas.multiEval.cc.layer.esFlatness.current.clone()
```

Subgroups

6.2.1.3.1.14 ScIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:CURRent:SCINdex
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.esFlatness.current.
↪scIndex.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns subcarrier indices of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. At these SC indices, the current minimum and maximum power of the equalizer coefficients have been detected within range 1 and range 2.

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

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

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

6.2.1.3.1.15 Extreme

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:ESFLatness:EXTReme
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:ESFLatness:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:EXTReme
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.esFlatness.extreme.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return current, average and extreme limit check results for the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:EXTReme
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.extreme.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:EXTReme
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.extreme.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

6.2.1.3.1.16 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:SDEViation
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness: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 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)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.standardDev.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:ESFLatness:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.esFlatness.standardDev.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return current, average, extreme and standard deviation single-value results of the equalizer spectrum flatness measurement, for carrier <no>, layer/antenna <l>. See also 'Equalizer spectrum flatness limits'.

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

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

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

6.2.1.3.1.17 EvMagnitude

class EvMagnitudeCls

EvMagnitude 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.nrSubMeas.multiEval.cc.layer.evMagnitude.clone()
```

Subgroups

6.2.1.3.1.18 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:EVMagnitude:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:EVMagnitude:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.
↪average.calculate(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, … See also 'Square EVM'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.average.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

6.2.1.3.1.19 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:EVMagnitude:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:EVMagnitude:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.
↪current.calculate(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.current.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

6.2.1.3.1.20 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:EVMagnitude:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:EVMagnitude:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:MAXimum
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.
↪maximum.calculate(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.maximum.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.maximum.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the EVM RMS diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

6.2.1.3.1.21 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.nrSubMeas.multiEval.cc.layer.evMagnitude.peak.clone()
```

Subgroups

6.2.1.3.1.22 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.peak.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.peak.
↪average.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

6.2.1.3.1.23 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.peak.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.peak.
↪current.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

6.2.1.3.1.24 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.peak.
↪maximum.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:EVMagnitude:PEAK:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.evMagnitude.peak.
↪maximum.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM peak diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. 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, ... See also 'Square EVM'.

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

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

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

6.2.1.3.1.25 Iemission

class IemissionCls

Iemission commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.3.1.26 Margin

class MarginCls

Margin 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.nrSubMeas.multiEval.cc.layer.iemission.margin.clone()
```

Subgroups

6.2.1.3.1.27 Average

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:AVERage
```

class AverageCls

Average 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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.iemission.margin.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Return the limit line margin results for carrier <no>, layer/antenna <l>. 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')

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

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.nrSubMeas.multiEval.cc.layer.iemission.margin.average.clone()
```

Subgroups

6.2.1.3.1.28 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:AVERage: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 for the general margin (at non-allocated RBs)
- Iq_Image: int: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: int: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:AVERage:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.iemission.margin.
↪average.rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪layer = repcap.Layer.Default)
```

Return resource block indices for in-band emission margins, for carrier <no>, layer/antenna <l>. At these RB indices, the CURRent, AVERage and EXTReme margins have been detected (see method RsCMPX_NrFr1Meas.NrSubMeas.MultiEval.Cc.Layer. Iemission.Margin.Current.fetch and ...:AVERage/EXTReme) .

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

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

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

6.2.1.3.1.29 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission: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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.iemission.margin.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Return the limit line margin results for carrier <no>, layer/antenna <l>. 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')

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

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.nrSubMeas.multiEval.cc.layer.iemission.margin.current.clone()
```

Subgroups

6.2.1.3.1.30 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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: '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 for the general margin (at non-allocated RBs)
- Iq_Image: int: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: int: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:CURRent:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.iemission.margin.
↪current.rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪layer = repcap.Layer.Default)
```

Return resource block indices for in-band emission margins, for carrier <no>, layer/antenna <l>. At these RB indices, the CURRent, AVERage and EXTReme margins have been detected (see method RsCMPX_NrFr1Meas.NrSubMeas.MultiEval.Cc.Layer. Iemission.Margin.Current.fetch and ...:AVERage/EXTReme) .

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

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

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

6.2.1.3.1.31 Extreme

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission: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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:EXTReme
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.iemission.margin.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Return the limit line margin results for carrier <no>, layer/antenna <l>. 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')

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

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.nrSubMeas.multiEval.cc.layer.iemission.margin.extreme.clone()
```

Subgroups

6.2.1.3.1.32 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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: '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 for the general margin (at non-allocated RBs)
- Iq_Image: int: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: int: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.iemission.margin.
↪extreme.rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪layer = repcap.Layer.Default)
```

Return resource block indices for in-band emission margins, for carrier <no>, layer/antenna <l>. At these RB indices, the CURRent, AVERage and EXTReme margins have been detected (see method RsCMPX_NrFr1Meas.NrSubMeas.MultiEval.Cc.Layer. Iemission.Margin.Current.fetch and ...:AVERage/EXTReme) .

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

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

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

6.2.1.3.1.33 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission: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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:IEMission:MARGin:SDEViation
value: FetchStruct = driver.nrSubMeas.multiEval.cc.layer.iemission.margin.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Return the limit line margin results for carrier <no>, layer/antenna <l>. 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')

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

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

6.2.1.3.1.34 Merror

class MerrorCls

Merror commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.3.1.35 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor: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]: Magnitude error value for low EVM window position.
- High: List[float]: Magnitude error value for high EVM window position.

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.merror.average.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.merror.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the magnitude error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.merror.average.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the magnitude error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

6.2.1.3.1.36 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor: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]: Magnitude error value for low EVM window position.
- High: List[float]: Magnitude error value for high EVM window position.

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.merror.current.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.merror.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the magnitude error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There

is one pair of magnitude error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.merror.current.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the magnitude error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

6.2.1.3.1.37 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MERRor: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]: Magnitude error value for low EVM window position.
- High: List[float]: Magnitude error value for high EVM window position.

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:MAXimum
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.merror.maximum.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.merror.maximum.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the magnitude error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MERRor:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.merror.maximum.
```

(continues on next page)

(continued from previous page)

```
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the magnitude error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of magnitude error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

6.2.1.3.1.38 Modulation

class ModulationCls

Modulation commands group definition. 11 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.3.1.39 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MODulation:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MODulation:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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
- Freq_Error_Ppm: float or bool: Carrier frequency error in ppm
- Sample_Clock_Err: float or bool: No parameter help available
- Antenna_1_Power: float: No parameter help available
- Antenna_2_Power: 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
- Freq_Error_Ppm: float: Carrier frequency error in ppm
- Sample_Clock_Err: float: No parameter help available
- Antenna_1_Power: float: No parameter help available
- Antenna_2_Power: float: No parameter help available

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.modulation.average.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.average.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

6.2.1.3.1.40 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MODulation:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MODulation:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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
- Freq_Error_Ppm: float or bool: Carrier frequency error in ppm
- Sample_Clock_Err: float or bool: No parameter help available
- Antenna_1_Power: float: No parameter help available
- Antenna_2_Power: 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
- Freq_Error_Ppm: float: Carrier frequency error in ppm
- Sample_Clock_Err: float: No parameter help available
- Antenna_1_Power: float: No parameter help available
- Antenna_2_Power: float: No parameter help available

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.modulation.current.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.current.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

6.2.1.3.1.41 Extreme

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MODulation:EXTReme
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:MODulation:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]: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
- 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
- Freq_Error_Ppm: float or bool: Carrier frequency error in ppm
- Sample_Clock_Err: float or bool: No parameter help available
- Ant_1_Pow_Min: float: Minimum power at antenna 1
- Ant_1_Pow_Max: float: Maximum power at antenna 1
- Ant_2_Pow_Min: float: Minimum power at antenna 2
- Ant_2_Pow_Max: float: Maximum power at antenna 2

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_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
- Freq_Error_Ppm: float: Carrier frequency error in ppm
- Sample_Clock_Err: float: No parameter help available
- Ant_1_Pow_Min: float: Minimum power at antenna 1
- Ant_1_Pow_Max: float: Maximum power at antenna 1
- Ant_2_Pow_Min: float: Minimum power at antenna 2
- Ant_2_Pow_Max: float: Maximum power at antenna 2

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:EXTReme
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.modulation.extreme.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return the extreme single value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:EXTReme
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.extreme.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return the extreme single value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:EXTReme
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.extreme.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return the extreme single value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

6.2.1.3.1.42 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:SDEViation
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation: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 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
- Freq_Error_Ppm: float: Carrier frequency error in ppm
- Sample_Clock_Err: float: No parameter help available
- Antenna_1_Power: float: No parameter help available
- Antenna_2_Power: float: No parameter help available

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.standardDev.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:MODulation:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.modulation.standardDev.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Return the current, average and standard deviation single-value results for carrier <no>, layer/antenna <l>. 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.

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

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

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

6.2.1.3.1.43 Perror

class PerrorCls

Perror commands group definition. 9 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.3.1.44 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor: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]: Phase error value for low EVM window position.
- High: List[float]: Phase error value for high EVM window position.

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.perror.average.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.perror.average.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the phase error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:AVERage
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.perror.average.
```

(continues on next page)

(continued from previous page)

```
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the phase error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

6.2.1.3.1.45 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor: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]: Phase error value for low EVM window position.
- High: List[float]: Phase error value for high EVM window position.

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.perror.current.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.perror.current.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the phase error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:CURRent
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.perror.current.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the phase error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

6.2.1.3.1.46 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:PERRor: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]: Phase error value for low EVM window position.
- High: List[float]: Phase error value for high EVM window position.

calculate(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:MAXimum
value: CalculateStruct = driver.nrSubMeas.multiEval.cc.layer.perror.maximum.
↪calculate(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

No command help available

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

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

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

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.perror.maximum.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the phase error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved.

There is one pair of phase error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:PERRor:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.cc.layer.perror.maximum.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the phase error diagrams for the OFDM symbols in the measured slot, for carrier <no>, layer/antenna <l>. The results of the current, average and maximum diagrams can be retrieved. There is one pair of phase error values per OFDM symbol, returned in the following order: <Reliability>, {<Low>, <High>}symbol 0, {<Low>, <High>}symbol 1, ... See also 'Squares Magnitude Error, Phase Error'.

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

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

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

6.2.1.3.1.47 ReferenceMarker

class ReferenceMarkerCls

ReferenceMarker 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.nrSubMeas.multiEval.cc.layer.referenceMarker.clone()
```

Subgroups

6.2.1.3.1.48 EvMagnitude

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:REFMarker:EVMagnitude
```

class EvMagnitudeCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:REFMarker:EVMagnitude
value: float = driver.nrSubMeas.multiEval.cc.layer.referenceMarker.evMagnitude.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage, carrierComponent␣
↪= repcap.CarrierComponent.Default, layer = repcap.Layer.Default)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

return
yvalue: Absolute y-value of the marker position

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.cc.layer.referenceMarker.evMagnitude.clone()
```

Subgroups

6.2.1.3.1.49 Peak

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:REFMarker:EVMagnitude:PEAK
```

class PeakCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:REFMarker:EVMagnitude:PEAK
value: float = driver.nrSubMeas.multiEval.cc.layer.referenceMarker.evMagnitude.
↪peak.fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage,␣
↪carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

return
yvalue: Absolute y-value of the marker position

6.2.1.3.1.50 Merror

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:REFMarker:MERRor
```

class MerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:REFMarker:MERRor
value: float = driver.nrSubMeas.multiEval.cc.layer.referenceMarker.merror.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage, carrierComponent␣
↪= repcap.CarrierComponent.Default, layer = repcap.Layer.Default)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

return
yvalue: Absolute y-value of the marker position

6.2.1.3.1.51 Perror

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>]:REFMarker:PERRor
```

class PerrorCls

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

fetch(*xvalue: int*, *trace_select: TraceSelect*, *carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LAYer<layer>
↪]:REFMarker:PERRor
value: float = driver.nrSubMeas.multiEval.cc.layer.referenceMarker.perror.
↪fetch(xvalue = 1, trace_select = enums.TraceSelect.AVERage, carrierComponent␣
↪= repcap.CarrierComponent.Default, layer = repcap.Layer.Default)
```

Uses the reference marker on the diagrams: EVM RMS, EVM peak, magnitude error and phase error vs OFDM symbol.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position There are two x-values per OFDM symbol on the x-axis (symbol 0 low, symbol 0 high, ..., symbol 14 low, symbol 14 high) .

param trace_select
No help available

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

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

return
yvalue: Absolute y-value of the marker position

6.2.1.3.2 Modulation

class ModulationCls

Modulation 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.nrSubMeas.multiEval.cc.modulation.clone()
```

Subgroups

6.2.1.3.2.1 Dallocation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>]: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 transmission bandwidth

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>
↪]:MODulation:DALLocation
value: FetchStruct = driver.nrSubMeas.multiEval.cc.modulation.dallocation.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the detected allocation for the measured slot.

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.

6.2.1.3.2.2 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>]:MODulation:DCHType
```

class DchTypeCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → ChannelTypeA

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>
↪]:MODulation:DCHType
value: enums.ChannelTypeA = driver.nrSubMeas.multiEval.cc.modulation.dchType.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the channel type for the measured slot, for carrier <no>.

Suppressed linked return values: reliability

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

return
channel_type: No help available

6.2.1.3.2.3 Dmodulation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>]:MODulation:DMODulation
```

class DmodulationCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → Modulation

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>
↪]:MODulation:DMODulation
value: enums.Modulation = driver.nrSubMeas.multiEval.cc.modulation.dmodulation.
↪fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Returns the detected modulation scheme in the measured slot, for carrier <no>.

Suppressed linked return values: reliability

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

return
modulation: BPSK, BPWS: /2-BPSK, /2-BPSK with shaping QPSK, Q16, Q64, Q256: QPSK, 16QAM, 64QAM, 256QAM

6.2.1.4 Dmarker<DeltaMarker>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.dmarker.repcap_deltaMarker_get()
driver.nrSubMeas.multiEval.dmarker.repcap_deltaMarker_set(repcap.DeltaMarker.Nr1)
```

class DmarkerCls

Dmarker commands group definition. 2 total commands, 2 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.nrSubMeas.multiEval.dmarker.clone()
```

Subgroups

6.2.1.4.1 Pdynamics

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:DMARker<No>:PDYNamics
value: float = driver.nrSubMeas.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
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

6.2.1.4.2 Pmonitor

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:DMARker<No>:PMONitor
```

class PmonitorCls

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

fetch(*xvalue: int*, *deltaMarker=DeltaMarker.Default*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:DMARker<No>:PMONitor
value: float = driver.nrSubMeas.multiEval.dmarker.pmonitor.fetch(xvalue = 1,␣
↪deltaMarker = repcap.DeltaMarker.Default)
```

Uses the markers 1 and 2 with relative values on the power monitor trace.

Suppressed linked return values: reliability

param xvalue
X-value of the marker position relative to the x-value of the reference marker (in slots)

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

6.2.1.5 Layer<Layer>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.layer.repcap_layer_get()
driver.nrSubMeas.multiEval.layer.repcap_layer_set(repcap.Layer.Nr1)
```

class LayerCls

Layer commands group definition. 14 total commands, 1 Subgroups, 0 group commands Repeated Capability: Layer, default value after init: Layer.Nr1

Cloning the Group

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

Subgroups

6.2.1.5.1 Pdynamics

class PdynamicsCls

Pdynamics commands group definition. 14 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.5.1.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics: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
- 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 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(*layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.layer.pdynamics.average.
↪calculate(layer = repcap.Layer.Default)
```

No command help available

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

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

fetch(*layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:AVERage
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.average.
↪fetch(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

read(*layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:AVERage
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.average.
↪read(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

6.2.1.5.1.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics: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
- 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 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(*layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.layer.pdynamics.current.
↪calculate(layer = repcap.Layer.Default)
```

No command help available

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

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

fetch(*layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:CURRent
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.current.
↪fetch(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

read(*layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:CURRent
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.current.
↪read(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

6.2.1.5.1.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics: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
- 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 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(*layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:MAXimum
value: CalculateStruct = driver.nrSubMeas.multiEval.layer.pdynamics.maximum.
↪calculate(layer = repcap.Layer.Default)
```

No command help available

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

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

fetch(*layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.maximum.
↪fetch(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

read(*layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.maximum.
↪read(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

6.2.1.5.1.4 Minimum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:MINimum
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:MINimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics: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
- 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 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(*layer=Layer.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:MINimum
value: CalculateStruct = driver.nrSubMeas.multiEval.layer.pdynamics.minimum.
↪calculate(layer = repcap.Layer.Default)
```

No command help available

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

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

fetch(*layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:MINimum
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.minimum.
↪fetch(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

read(*layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:MINimum
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.minimum.
↪read(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

6.2.1.5.1.5 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics:SDEViation
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>]:PDYNamics: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 dynamics 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

fetch(*layer=Layer.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.standardDev.
↪fetch(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

read(*layer=Layer.Default*) → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer>
↪]:PDYNamics:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.layer.pdynamics.standardDev.
↪read(layer = repcap.Layer.Default)
```

Return the single-value results of the power dynamics measurement. The current, average, minimum, maximum and standard deviation results can be retrieved. The OFF power results refer to antenna <l>. The ON power results refer to the sum of both antenna signals.

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

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

6.2.1.6 ListPy

class ListPyCls

ListPy commands group definition. 354 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.6.1 Aclr

class AclrCls

Aclr commands group definition. 34 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.6.1.1 Dallocation

SCPI Command :

```
FETCh:NRSub: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 transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:DALLocation
value: FetchStruct = driver.nrSubMeas.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 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.

6.2.1.6.1.2 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:DCHType
```

class DchTypeCls

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

fetch() → List[ChannelTypeA]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:DCHType
value: List[enums.ChannelTypeA] = driver.nrSubMeas.multiEval.listPy.aclr.
↪dchType.fetch()
```

No command help available

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.6.1.3 Endc

class EndcCls

Endc 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.nrSubMeas.multiEval.listPy.aclr.endc.clone()
```

Subgroups

6.2.1.6.1.4 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:AVERage
```

class AverageCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.endc.
↪average.calculate()
```

Return the power in the allocated NR channel for EN-DC mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.endc.average.fetch()
```

Return the power in the allocated NR channel for EN-DC mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

6.2.1.6.1.5 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:CURRent
```

class CurrentCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.endc.
↪current.calculate()
```

Return the power in the allocated NR channel for EN-DC mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.endc.current.fetch()
```

Return the power in the allocated NR channel for EN-DC mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

6.2.1.6.1.6 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.nrSubMeas.multiEval.listPy.aclr.endc.negativ.clone()
```

Subgroups

6.2.1.6.1.7 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:NEGativ:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:NEGativ:AVERage
```

class AverageCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:NEGativ:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.endc.
↪negativ.average.calculate()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:NEGativ:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.endc.negativ.
↪average.fetch()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

6.2.1.6.1.8 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:NEGativ:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:NEGativ:CURRent
```

class CurrentCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:NEGativ:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.endc.
↪negativ.current.calculate()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:NEGativ:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.endc.negativ.
↪current.fetch()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

6.2.1.6.1.9 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.nrSubMeas.multiEval.listPy.aclr.endc.positiv.clone()
```

Subgroups

6.2.1.6.1.10 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:POSitiv:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:POSitiv:AVERage
```

class AverageCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:POSitiv:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.endc.
↪positiv.average.calculate()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:POSitiv:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.endc.positiv.
↪average.fetch()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

6.2.1.6.1.11 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:POSitiv:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:ENDC:POSitiv:CURRent
```

class CurrentCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:POSitiv:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.endc.
↪positiv.current.calculate()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:ENDC:POSitiv:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.endc.positiv.
↪current.fetch()
```

Return the ACLR for the adjacent channel above (POSitiv) or below (NEGativ) the carrier frequency for EN-DC mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

6.2.1.6.1.12 Nr

class NrCls

Nr 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.nrSubMeas.multiEval.listPy.aclr.nr.clone()
```

Subgroups

6.2.1.6.1.13 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:AVERage
```

class AverageCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.nr.
↪average.calculate()
```

Return the power in the allocated NR channel for NR standalone mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.nr.average.fetch()
```

Return the power in the allocated NR channel for NR standalone mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

6.2.1.6.1.14 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:CURRent
```

class CurrentCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.nr.
↪current.calculate()
```

Return the power in the allocated NR channel for NR standalone mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.nr.current.fetch()
```

Return the power in the allocated NR channel for NR standalone mode, for all measured list mode 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
carrier: Comma-separated list of values, one per measured segment

6.2.1.6.1.15 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.nrSubMeas.multiEval.listPy.aclr.nr.negativ.clone()
```

Subgroups

6.2.1.6.1.16 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:NEGativ:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:NEGativ:AVERage
```

class AverageCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:NEGativ:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.nr.
↪negativ.average.calculate()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:NEGativ:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.nr.negativ.average.
↪fetch()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

6.2.1.6.1.17 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:NEGativ:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:NEGativ:CURRent
```

class CurrentCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:NEGativ:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.nr.
↪negativ.current.calculate()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:NEGativ:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.nr.negativ.current.
↪fetch()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_negativ: Comma-separated list of values, one per measured segment

6.2.1.6.1.18 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.nrSubMeas.multiEval.listPy.aclr.nr.positiv.clone()
```

Subgroups

6.2.1.6.1.19 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:POSitiv:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:POSitiv:AVERage
```

class AverageCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:POSitiv:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.nr.
↪positiv.average.calculate()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:POSitiv:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.nr.positiv.average.
↪fetch()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

6.2.1.6.1.20 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:POSitiv:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:NR:POSitiv:CURRent
```

class CurrentCls

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

calculate() → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:POSitiv:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.nr.
↪positiv.current.calculate()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:ACLR:NR:POSitiv:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.nr.positiv.current.
↪fetch()
```

Return the ACLR for the first adjacent NR channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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
nr_positiv: Comma-separated list of values, one per measured segment

6.2.1.6.1.21 Utra<UtraChannel>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.aclr.utra.repcap_utraChannel_get()
driver.nrSubMeas.multiEval.listPy.aclr.utra.repcap_utraChannel_set(repcap.UtraChannel.
↪Nr1)
```

class UtraCls

Utra commands group definition. 8 total commands, 2 Subgroups, 0 group commands Repeated Capability: UtraChannel, default value after init: UtraChannel.Nr1

Cloning the Group

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

Subgroups

6.2.1.6.1.22 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.nrSubMeas.multiEval.listPy.aclr.utra.negativ.clone()
```

Subgroups

6.2.1.6.1.23 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:NEGativ:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:NEGativ:AVERage
```

class AverageCls

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

calculate(*utraChannel=UtraChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:NEGativ:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.utra.
↪negativ.average.calculate(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

fetch(*utraChannel=UtraChannel.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:NEGativ:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.utra.negativ.
↪average.fetch(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

6.2.1.6.1.24 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:NEGativ:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:NEGativ:CURRent
```

class CurrentCls

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

calculate(*utraChannel=UtraChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:NEGativ:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.utra.
↪negativ.current.calculate(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

fetch(*utraChannel=UtraChannel.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:NEGativ:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.utra.negativ.
↪current.fetch(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_negativ: Comma-separated list of values, one per measured segment

6.2.1.6.1.25 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.nrSubMeas.multiEval.listPy.aclr.utra.positiv.clone()
```

Subgroups

6.2.1.6.1.26 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:POSitiv:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:POSitiv:AVERage
```

class AverageCls

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

calculate(*utraChannel=UtraChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:POSitiv:AVERage
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.utra.
↪positiv.average.calculate(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

fetch(*utraChannel=UtraChannel.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:POSitiv:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.utra.positiv.
↪average.fetch(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

6.2.1.6.1.27 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:POSitiv:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>:POSitiv:CURRent
```

class CurrentCls

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

calculate(*utraChannel=UtraChannel.Default*) → List[ResultStatus2]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:POSitiv:CURRent
value: List[enums.ResultStatus2] = driver.nrSubMeas.multiEval.listPy.aclr.utra.
↪positiv.current.calculate(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

fetch(*utraChannel=UtraChannel.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:ACLR:UTRA<nr6g>
↪:POSitiv:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.aclr.utra.positiv.
↪current.fetch(utraChannel = repcap.UtraChannel.Default)
```

Return the ACLR for the first or second adjacent UTRA channel above (POSitiv) or below (NEGativ) the carrier frequency for NR standalone mode, for all measured list mode 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 utraChannel
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Utra')

return
utra_positiv: Comma-separated list of values, one per measured segment

6.2.1.6.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.cc.repcap_carrierComponent_get()
driver.nrSubMeas.multiEval.listPy.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

class CcCls

Cc commands group definition. 214 total commands, 3 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.nrSubMeas.multiEval.listPy.cc.clone()
```

Subgroups

6.2.1.6.2.1 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.nrSubMeas.multiEval.listPy.cc.esFlatness.clone()
```

Subgroups

6.2.1.6.2.2 Difference<Difference>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.difference.repcap_difference_get()
driver.nrSubMeas.multiEval.listPy.cc.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.nrSubMeas.multiEval.listPy.cc.esFlatness.difference.clone()
```

Subgroups

6.2.1.6.2.3 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:DIFFerence
↪<nr6g>:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:DIFFerence<nr6g>:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:DIFFerence<nr6g>:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪difference.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default, difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:DIFFerence<nr6g>:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.difference.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, difference␣
↪= repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.4 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:DIFFerence
<nr6g>:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
]:ESFLatness:DIFFerence<nr6g>:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
]:ESFLatness:DIFFerence<nr6g>:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
difference.current.calculate(carrierComponent = repcap.CarrierComponent.
Default, difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
]:ESFLatness:DIFFerence<nr6g>:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.difference.
current.fetch(carrierComponent = repcap.CarrierComponent.Default, difference␣
= repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.5 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:DIFFerence
↪<nr6g>:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:DIFFerence<nr6g>:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *difference=Difference.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:DIFFerence<nr6g>:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪difference.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default, difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:DIFFerence<nr6g>:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.difference.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default, difference␣
↪= repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.6 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:DIFFerence
↪<nr6g>:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*, *difference=Difference.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:DIFFerence<nr6g>:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.difference.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪difference = repcap.Difference.Default)
```

Return equalizer spectrum flatness single value results (differences between ranges) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.7 Maxr<MaxRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.maxr.repcap_maxRange_get()
driver.nrSubMeas.multiEval.listPy.cc.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.nrSubMeas.multiEval.listPy.cc.esFlatness.maxr.clone()
```

Subgroups

6.2.1.6.2.8 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MAXR<nr6g>
↪:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MAXR
↪<nr6g>:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MAXR<nr6g>:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪maxr.average.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MAXR<nr6g>:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.maxr.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, maxRange =␣
↪repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.9 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MAXR<nr6g>
↪:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MAXR
↪<nr6g>:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MAXR<nr6g>:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪maxr.current.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MAXR<nr6g>:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.maxr.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default, maxRange =␣
↪repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.10 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MAXR<nr6g>
↪:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MAXR
↪<nr6g>:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *maxRange=MaxRange.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MAXR<nr6g>:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪maxr.extreme.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MAXR<nr6g>:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.maxr.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default, maxRange =␣
↪repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.11 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MAXR<nr6g>
↪:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*, *maxRange=MaxRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MAXR<nr6g>:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.maxr.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪maxRange = repcap.MaxRange.Default)
```

Return equalizer spectrum flatness single value results (maximum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.12 Minr<MinRange>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.minr.repcap_minRange_get()
driver.nrSubMeas.multiEval.listPy.cc.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.nrSubMeas.multiEval.listPy.cc.esFlatness.minr.clone()
```

Subgroups

6.2.1.6.2.13 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MINR<nr6g>
↪:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MINR
↪<nr6g>:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MINR<nr6g>:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪minr.average.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MINR<nr6g>:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.minr.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, minRange =␣
↪repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.

6.2.1.6.2.14 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MINR<nr6g>
↪:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MINR
↪<nr6g>:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MINR<nr6g>:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪minr.current.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MINR<nr6g>:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.minr.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default, minRange =␣
↪repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.

6.2.1.6.2.15 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MINR<nr6g>
↪:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MINR
↪<nr6g>:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *minRange=MinRange.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MINR<nr6g>:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪minr.extreme.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MINR<nr6g>:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.minr.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default, minRange =␣
↪repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.

6.2.1.6.2.16 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:MINR<nr6g>
↪:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*, *minRange=MinRange.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:MINR<nr6g>:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.minr.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪minRange = repcap.MinRange.Default)
```

Return equalizer spectrum flatness single value results (minimum within a range) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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.

6.2.1.6.2.17 Ripple<Ripple>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.ripple.repcap_ripple_get()
driver.nrSubMeas.multiEval.listPy.cc.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.nrSubMeas.multiEval.listPy.cc.esFlatness.ripple.clone()
```

Subgroups

6.2.1.6.2.18 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:RIPPle<nr6g>
↪:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:RIPPle
↪<nr6g>:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:RIPPle<nr6g>:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪ripple.average.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:RIPPle<nr6g>:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.ripple.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, ripple =␣
↪repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.19 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:RIPPle<nr6g>
↪:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:RIPPle
↪<nr6g>:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:RIPPle<nr6g>:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪ripple.current.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:RIPPle<nr6g>:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.ripple.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default, ripple =␣
↪repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.20 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:RIPPle<nr6g>
↪:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:RIPPle
↪<nr6g>:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*, *ripple=Ripple.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:RIPPle<nr6g>:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.
↪ripple.extreme.calculate(carrierComponent = repcap.CarrierComponent.Default,␣
↪ripple = repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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(*carrierComponent=CarrierComponent.Default*, *ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:RIPPle<nr6g>:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.ripple.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default, ripple =␣
↪repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.21 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:ESFLatness:RIPPle<nr6g>
↪:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*, *ripple=Ripple.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:RIPPle<nr6g>:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.ripple.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default, ripple␣
↪= repcap.Ripple.Default)
```

Return equalizer spectrum flatness single value results (ripple 1 or ripple 2) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

6.2.1.6.2.22 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.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.clone()
```

Subgroups

6.2.1.6.2.23 Maximum<Maximum>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.maximum.repcap_maximum_get()
driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.maximum.repcap_maximum_
↪set(repcap.Maximum.Nr1)
```

class MaximumCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.maximum.clone()
```

Subgroups

6.2.1.6.2.24 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:SCINdex:MAXimum<nr6g>:CURRent
```

class CurrentCls

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

fetch(*carrierComponent=CarrierComponent.Default*, *maximum=Maximum.Default*) → List[int]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:SCINdex:MAXimum<nr6g>:CURRent
value: List[int] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.
↪maximum.current.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪maximum = repcap.Maximum.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

return
maximum: Comma-separated list of values, one per measured segment

6.2.1.6.2.25 Minimum<Minimum>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.minimum.repcap_minimum_get()
driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.minimum.repcap_minimum_
↪set(repcap.Minimum.Nr1)
```

class MinimumCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.minimum.clone()
```

Subgroups

6.2.1.6.2.26 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:SCINdex:MINimum<nr6g>:CURRent
```

class CurrentCls

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

fetch(*carrierComponent=CarrierComponent.Default*, *minimum=Minimum.Default*) → List[int]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:ESFLatness:SCINdex:MINimum<nr6g>:CURRent
value: List[int] = driver.nrSubMeas.multiEval.listPy.cc.esFlatness.scIndex.
↪minimum.current.fetch(carrierComponent = repcap.CarrierComponent.Default,␣
↪minimum = repcap.Minimum.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

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

return
minimum: Comma-separated list of values, one per measured segment

6.2.1.6.2.27 Iemission

class IemissionCls

Iemission commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.6.2.28 Margin

class MarginCls

Margin 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.nrSubMeas.multiEval.listPy.cc.iemission.margin.clone()
```

Subgroups

6.2.1.6.2.29 Average

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:IEMission:MARGin:AVERage
```

class AverageCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Margin: List[float]: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: List[float]: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)

- Carr_Leakage: List[float]: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.iemission.margin.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for all measured list mode segments, for carrier <c>. 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. The results are returned as triplets per segment: <Reliability>, {<Margin>, <IQImage>, <CarrLeakage>}seg 1, {<Margin>, <IQImage>, <CarrLeakage>}seg 2, ...

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.nrSubMeas.multiEval.listPy.cc.iemission.margin.average.clone()
```

Subgroups

6.2.1.6.2.30 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:AVERage:RBINdex
```

class RbIndexCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rb_Index: List[int]: Resource block index for the general margin (at non-allocated RBs)
- Iq_Image: List[int]: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: List[int]: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:AVERage:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.iemission.margin.
↪average.rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the in-band emission measurement for all measured list mode segments, for carrier <c>. At these RB indices, the CURRent, AVERage and EXTReme margins have been detected. The results are returned as triplets per segment: <Reliability>, {<RBindex>, <IQImage>, <CarrLeakage>}seg 1, {<RBindex>, <IQImage>, <CarrLeakage>}seg 2, ...

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.

6.2.1.6.2.31 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:IEMission: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'
- Margin: List[float]: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: List[float]: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: List[float]: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.iemission.margin.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for all measured list mode segments, for carrier <c>. 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. The results are returned as triplets per segment: <Reliability>, {<Margin>, <IQImage>, <CarrLeakage>}seg 1, {<Margin>, <IQImage>, <CarrLeakage>}seg 2, ...

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.nrSubMeas.multiEval.listPy.cc.iemission.margin.current.clone()
```

Subgroups

6.2.1.6.2.32 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]: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: 'Reliability indicator'
- Rb_Index: List[int]: Resource block index for the general margin (at non-allocated RBs)
- Iq_Image: List[int]: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: List[int]: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:CURRent:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.iemission.margin.
↪current.rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the in-band emission measurement for all measured list mode segments, for carrier <c>. At these RB indices, the CURRent, AVERage and EXTReme margins have been detected. The results are returned as triplets per segment: <Reliability>, {<RBindex>, <IQImage>, <CarrLeakage>}seg 1, {<RBindex>, <IQImage>, <CarrLeakage>}seg 2, ...

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.

6.2.1.6.2.33 Extreme

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:IEMission: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'
- Margin: List[float]: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: List[float]: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: List[float]: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:EXTReme
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.iemission.margin.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for all measured list mode segments, for carrier <c>. 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. The results are returned as triplets per segment: <Reliability>, {<Margin>, <IQImage>, <CarrLeakage>}seg 1, {<Margin>, <IQImage>, <CarrLeakage>}seg 2, ...

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.nrSubMeas.multiEval.listPy.cc.iemission.margin.extreme.clone()
```

Subgroups

6.2.1.6.2.34 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]: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: 'Reliability indicator'
- Rb_Index: List[int]: Resource block index for the general margin (at non-allocated RBs)
- Iq_Image: List[int]: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: List[int]: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.iemission.margin.
↪extreme.rbIndex.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the in-band emission measurement for all measured list mode segments, for carrier <c>. At these RB indices, the CURRent, AVERage and EXTReme margins have been detected. The results are returned as triplets per segment: <Reliability>, {<RBindex>, <IQImage>, <CarrLeakage>}seg 1, {<RBindex>, <IQImage>, <CarrLeakage>}seg 2, ...

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.

6.2.1.6.2.35 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission: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'
- Margin: List[float]: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: List[float]: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: List[float]: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:IEMission:MARGin:SDEViation
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.iemission.margin.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for all measured list mode segments, for carrier <c>. 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. The results are returned as triplets per segment: <Reliability>, {<Margin>, <IQImage>, <CarrLeakage>}seg 1, {<Margin>, <IQImage>, <CarrLeakage>}seg 2, ...

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.

6.2.1.6.2.36 Modulation

class ModulationCls

Modulation commands group definition. 177 total commands, 12 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1.6.2.37 Dallocation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]: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 transmission bandwidth

fetch(*carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:DALLocation
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪dallocation.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

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 results are returned as pairs per segment: <Reliability>, {<NrResBlocks>, <OffsetResBlocks>}seg 1, {<NrResBlocks>, <OffsetResBlocks>}seg 2, ...

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.

6.2.1.6.2.38 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:MODulation:DCHType
```

class DchTypeCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[ChannelTypeA]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:DCHType
value: List[enums.ChannelTypeA] = driver.nrSubMeas.multiEval.listPy.cc.
↪modulation.dchType.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

No command help available

Suppressed linked return values: reliability

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

return
channel_type: No help available

6.2.1.6.2.39 Dmodulation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:MODulation:DMODulation
```

class DmodulationCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[Modulation]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:DMODulation
value: List[enums.Modulation] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪dmodulation.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return the detected modulation scheme for all measured list mode segments, for carrier <c>. The result is determined from the last measured slot of the statistical length of a segment.

Suppressed linked return values: reliability

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

return
modulation: Comma-separated list of values, one per measured segment BPSK, BPWS: /2-BPSK, /2-BPSK with shaping QPSK, Q16, Q64, Q256: QPSK, 16QAM, 64QAM, 256QAM

6.2.1.6.2.40 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.clone()
```

Subgroups

6.2.1.6.2.41 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.clone()
```

Subgroups

6.2.1.6.2.42 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.high.clone()
```

Subgroups

6.2.1.6.2.43 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.dmrs.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.44 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.dmrs.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.45 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.dmrs.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.46 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.47 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.low.clone()
```

Subgroups

6.2.1.6.2.48 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.dmrs.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.49 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.dmrs.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.50 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.dmrs.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.51 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:DMRS:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.dmrs.
↪low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.52 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.clone()
```

Subgroups

6.2.1.6.2.53 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.high.clone()
```

Subgroups

6.2.1.6.2.54 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.peak.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.55 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.peak.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.56 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
```

(continues on next page)

(continued from previous page)

```
↪evm.peak.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.57 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.58 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.low.clone()
```

Subgroups

6.2.1.6.2.59 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.peak.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.60 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.peak.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.61 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.peak.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.62 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:PEAK:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.peak.
↪low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.63 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.clone()
```

Subgroups

6.2.1.6.2.64 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.high.clone()
```

Subgroups

6.2.1.6.2.65 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.rms.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.66 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.rms.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.67 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.rms.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.68 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.69 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.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.low.clone()
```

Subgroups

6.2.1.6.2.70 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.rms.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.71 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.rms.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.72 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪evm.rms.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.73 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:SDEViation
```

class StandardDevCls

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

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:EVM:RMS:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.evm.rms.
↪low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return error vector magnitude RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
evm_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.74 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.nrSubMeas.multiEval.listPy.cc.modulation.freqError.clone()
```

Subgroups

6.2.1.6.2.75 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:AVERage
```

class AverageCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪freqError.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return carrier frequency error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
frequency_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.freqError.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return carrier frequency error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.76 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:CURRent
```

class CurrentCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪freqError.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return carrier frequency error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
frequency_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.freqError.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return carrier frequency error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.77 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:EXTReme
```

class ExtremeCls

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

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪freqError.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return carrier frequency error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
frequency_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.freqError.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return carrier frequency error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.78 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:FERRor:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.freqError.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return carrier frequency error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
frequency_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.79 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.nrSubMeas.multiEval.listPy.cc.modulation.iqOffset.clone()
```

Subgroups

6.2.1.6.2.80 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪iqOffset.average.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return I/Q origin offset values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
iq_offset: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.iqOffset.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return I/Q origin offset values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.6.2.81 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪iqOffset.current.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return I/Q origin offset values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
iq_offset: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.iqOffset.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return I/Q origin offset values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.6.2.82 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪iqOffset.extreme.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return I/Q origin offset values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
iq_offset: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:EXTReme
```

(continues on next page)

(continued from previous page)

```
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.iqOffset.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return I/Q origin offset values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.6.2.83 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:IQOFfset:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.iqOffset.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return I/Q origin offset values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
iq_offset: Comma-separated list of values, one per measured segment

6.2.1.6.2.84 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.clone()
```

Subgroups

6.2.1.6.2.85 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.dmrs.clone()
```

Subgroups

6.2.1.6.2.86 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.dmrs.high.clone()
```

Subgroups

6.2.1.6.2.87 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.dmrs.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.88 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
```

(continues on next page)

(continued from previous page)

```
↪merror.dmrs.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.89 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.dmrs.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.90 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.91 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.dmrs.low.clone()
```

Subgroups

6.2.1.6.2.92 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.dmrs.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.93 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.dmrs.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.94 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.dmrs.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.95 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:DMRS:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪dmrs.low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.96 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.peak.clone()
```

Subgroups

6.2.1.6.2.97 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.peak.high.clone()
```

Subgroups

6.2.1.6.2.98 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.peak.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.99 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.peak.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.100 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.peak.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.101 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_err_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.102 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.peak.low.clone()
```

Subgroups

6.2.1.6.2.103 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.peak.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.104 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.peak.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.105 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.peak.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.106 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:PEAK:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.
↪peak.low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.107 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.clone()
```

Subgroups

6.2.1.6.2.108 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.high.clone()
```

Subgroups

6.2.1.6.2.109 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.rms.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.110 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.rms.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.111 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
```

(continues on next page)

(continued from previous page)

```
↪merror.rms.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.112 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.113 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.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.low.clone()
```

Subgroups

6.2.1.6.2.114 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
 ↪]:MODulation:MERRor:RMS:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
 ↪]:MODulation:MERRor:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
 ↪]:MODulation:MERRor:RMS:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
 ↪merror.rms.low.average.calculate(carrierComponent = repcap.CarrierComponent.
 ↪Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.115 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.rms.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.116 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪merror.rms.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.117 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:MERRor:RMS:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.merror.rms.
↪low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return magnitude error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
mag_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.118 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.clone()
```

Subgroups

6.2.1.6.2.119 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.dmrs.clone()
```

Subgroups

6.2.1.6.2.120 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.dmrs.high.clone()
```

Subgroups

6.2.1.6.2.121 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.dmrs.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.122 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
```

(continues on next page)

(continued from previous page)

```
↪perror.dmrs.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.123 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.dmrs.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.124 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.125 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.dmrs.low.clone()
```

Subgroups

6.2.1.6.2.126 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.dmrs.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.127 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.dmrs.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.128 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.dmrs.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.129 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:DMRS:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪dmrs.low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error DMRS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_dmrs_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.130 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.peak.clone()
```

Subgroups

6.2.1.6.2.131 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.peak.high.clone()
```

Subgroups

6.2.1.6.2.132 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.peak.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.133 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.peak.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.134 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.peak.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.135 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.136 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.peak.low.clone()
```

Subgroups

6.2.1.6.2.137 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.peak.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.138 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.peak.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.139 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.peak.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.140 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:PEAK:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.
↪peak.low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error peak values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_peak_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.141 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.clone()
```

Subgroups

6.2.1.6.2.142 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.high.clone()
```

Subgroups

6.2.1.6.2.143 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.rms.high.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪high.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.144 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.rms.high.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪high.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.145 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
```

(continues on next page)

(continued from previous page)

```
↪perror.rms.high.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_high: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪high.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.146 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:HIGH:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪high.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_high: Comma-separated list of values, one per measured segment

6.2.1.6.2.147 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.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.low.clone()
```

Subgroups

6.2.1.6.2.148 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.rms.low.average.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪low.average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.149 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.rms.low.current.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪low.current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.150 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪perror.rms.low.extreme.calculate(carrierComponent = repcap.CarrierComponent.
↪Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_low: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪low.extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.151 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PERRor:RMS:LOW:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.perror.rms.
↪low.standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return phase error RMS values for low and high EVM window position, for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
ph_error_rms_low: Comma-separated list of values, one per measured segment

6.2.1.6.2.152 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.nrSubMeas.multiEval.listPy.cc.modulation.ppower.clone()
```

Subgroups

6.2.1.6.2.153 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪ppower.average.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.ppower.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.154 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪ppower.current.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.ppower.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.155 Maximum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MAXimum
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪ppower.maximum.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.ppower.
↪maximum.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.156 Minimum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MINimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MINimum
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪ppower.minimum.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:MINimum
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.ppower.
↪minimum.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.157 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PPOWer:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.ppower.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment peak power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
peak_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.158 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.nrSubMeas.multiEval.listPy.cc.modulation.psd.clone()
```

Subgroups

6.2.1.6.2.159 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:MODulation:PSD:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪psd.average.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.psd.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: Comma-separated list of values, one per measured segment

6.2.1.6.2.160 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:MODulation:PSD:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪psd.current.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.psd.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: Comma-separated list of values, one per measured segment

6.2.1.6.2.161 Maximum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:MODulation:PSD:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:MAXimum
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪psd.maximum.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.psd.
↪maximum.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: Comma-separated list of values, one per measured segment

6.2.1.6.2.162 Minimum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>]:MODulation:PSD:MINimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:MINimum
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪psd.minimum.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:MINimum
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.psd.
↪minimum.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: Comma-separated list of values, one per measured segment

6.2.1.6.2.163 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:PSD:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.psd.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return RB power values (power spectral density) for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
psd: Comma-separated list of values, one per measured segment

6.2.1.6.2.164 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.nrSubMeas.multiEval.listPy.cc.modulation.terror.clone()
```

Subgroups

6.2.1.6.2.165 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪terror.average.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return transmit time error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
timing_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.terror.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return transmit time error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.166 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪terror.current.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return transmit time error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
timing_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.terror.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return transmit time error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.167 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:EXTReme
```

class ExtremeCls

Extreme commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:EXTReme
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪terror.extreme.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return transmit time error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
timing_error: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:EXTReme
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.terror.
↪extreme.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return transmit time error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.168 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TERRor:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.terror.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return transmit time error values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
timing_error: Comma-separated list of values, one per measured segment

6.2.1.6.2.169 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.nrSubMeas.multiEval.listPy.cc.modulation.tpower.clone()
```

Subgroups

6.2.1.6.2.170 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:AVERage
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪tpower.average.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:AVERage
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.tpower.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.171 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:CURRent
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪tpower.current.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:CURRent
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.tpower.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.172 Maximum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MAXimum
```

(continues on next page)

(continued from previous page)

```
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪tpower.maximum.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.tpower.
↪maximum.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.173 Minimum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MINimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MINimum
```

class MinimumCls

Minimum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

calculate(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MINimum
value: List[float or bool] = driver.nrSubMeas.multiEval.listPy.cc.modulation.
↪tpower.minimum.calculate(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: (float or boolean items) Comma-separated list of values, one per measured segment

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:MINimum
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.tpower.
↪minimum.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.6.2.174 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*carrierComponent=CarrierComponent.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST[:CC<Carrier>
↪]:MODulation:TPOWer:SDEViation
value: List[float] = driver.nrSubMeas.multiEval.listPy.cc.modulation.tpower.
↪standardDev.fetch(carrierComponent = repcap.CarrierComponent.Default)
```

Return user equipment power values for all measured list mode segments, for carrier <c>. 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 carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

return
tx_power: Comma-separated list of values, one per measured segment

6.2.1.6.3 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.nrSubMeas.multiEval.listPy.pmonitor.clone()
```

Subgroups

6.2.1.6.3.1 Peak

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIST:PMONitor:PEAK
value: List[float] = driver.nrSubMeas.multiEval.listPy.pmonitor.peak.fetch()
```

Return the power monitor vs subframe 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_NrFr1Meas.NrSubMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Start.fetch
- method RsCMPX_NrFr1Meas.NrSubMeas.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.

6.2.1.6.3.2 Rms

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIST:PMONitor:RMS
value: List[float] = driver.nrSubMeas.multiEval.listPy.pmonitor.rms.fetch()
```

Return the power monitor vs subframe 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_NrFr1Meas.NrSubMeas.MultiEval.ListPy.Segment.Pmonitor.Array.Start.fetch
- method RsCMPX_NrFr1Meas.NrSubMeas.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.

6.2.1.6.3.3 Slots

class SlotsCls

Slots 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.nrSubMeas.multiEval.listPy.pmonitor.slots.clone()
```

Subgroups

6.2.1.6.3.4 Peak

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:PMONitor:SLOTs:PEAK
```

class PeakCls

Peak commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:PMONitor:SLOTs:PEAK
value: List[float] = driver.nrSubMeas.multiEval.listPy.pmonitor.slots.peak.
↪fetch()
```

Return the power monitor vs slot results for all measured segments in list mode. The commands return one power result per slot for the measured carrier. The power values are RMS averaged over the slot or represent the peak value within the slot.

INTRO_CMD_HELP: Commands for querying the result list structure:

- method RsCMPX_NrFr1Meas.NrSubMeas.MultiEval.ListPy.Segment.Pmonitor.Slots.Array.Start.fetch
- method RsCMPX_NrFr1Meas.NrSubMeas.MultiEval.ListPy.Segment.Pmonitor.Slots.Array.Length.fetch

Suppressed linked return values: reliability

return
step_peak_power: Comma-separated list of power values, one value per slot, from first slot of first measured segment to last slot of last measured segment For an inactive segment, only one INV is returned, independent of the number of slots.

6.2.1.6.3.5 Rms

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:PMONitor:SLOTs:RMS
```

class RmsCls

Rms commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:PMONitor:SLOTs:RMS
value: List[float] = driver.nrSubMeas.multiEval.listPy.pmonitor.slots.rms.
↪fetch()
```

Return the power monitor vs slot results for all measured segments in list mode. The commands return one power result per slot for the measured carrier. The power values are RMS averaged over the slot or represent the peak value within the slot.

INTRO_CMD_HELP: Commands for querying the result list structure:

- method RsCMPX_NrFr1Meas.NrSubMeas.MultiEval.ListPy.Segment.Pmonitor.Slots.Array.Start.fetch
- method RsCMPX_NrFr1Meas.NrSubMeas.MultiEval.ListPy.Segment.Pmonitor.Slots.Array.Length.fetch

Suppressed linked return values: reliability

return
step_rms_power: Comma-separated list of power values, one value per slot, from first slot of first measured segment to last slot of last measured segment For an inactive segment, only one INV is returned, independent of the number of slots.

6.2.1.6.4 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.nrSubMeas.multiEval.listPy.power.clone()
```

Subgroups

6.2.1.6.4.1 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.nrSubMeas.multiEval.listPy.power.txPower.clone()
```

Subgroups

6.2.1.6.4.2 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:AVERage
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:AVERage
value: List[float] = driver.nrSubMeas.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

6.2.1.6.4.3 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:CURRent
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:CURRent
value: List[float] = driver.nrSubMeas.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

6.2.1.6.4.4 Maximum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:MAXimum
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MAXimum
value: List[float] = driver.nrSubMeas.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

6.2.1.6.4.5 Minimum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:MINimum
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:POWer:TXPower:MINimum
value: List[float] = driver.nrSubMeas.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

6.2.1.6.4.6 StandardDev

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:POWer:TXPower:SDEViation
value: List[float] = driver.nrSubMeas.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

6.2.1.6.5 Segment

RepCap Settings

```
# Range: Nr1 .. Nr512
rc = driver.nrSubMeas.multiEval.listPy.segment.repcap_sEGMent_get()
driver.nrSubMeas.multiEval.listPy.segment.repcap_sEGMent_set(repcap.SEGMent.Nr1)
```

class SegmentCls

Segment commands group definition. 68 total commands, 5 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.nrSubMeas.multiEval.listPy.segment.clone()
```

Subgroups

6.2.1.6.5.1 Aclr

class AclrCls

Aclr 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.nrSubMeas.multiEval.listPy.segment.aclr.clone()
```

Subgroups

6.2.1.6.5.2 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:ACLR:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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
- Nr_Neg: enums.ResultStatus2: ACLR for the first NR channel with lower frequency
- Carrier: enums.ResultStatus2: Power in the allocated NR channel
- Nr_Pos: enums.ResultStatus2: ACLR for the first NR 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 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
- Nr_Neg: float: ACLR for the first NR channel with lower frequency
- Carrier: float: Power in the allocated NR channel
- Nr_Pos: float: ACLR for the first NR 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(*sEGMent=SEGMent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:ACLR:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.average.
↪calculate(sEGMent = repcap.SEGMent.Default)
```

Return ACLR single value results for NR standalone mode, list mode segment <no>. 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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:ACLR:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.average.
↪fetch(sEGMent = repcap.SEGMent.Default)
```

Return ACLR single value results for NR standalone mode, list mode segment <no>. 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.

6.2.1.6.5.3 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:ACLR:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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
- Nr_Neg: enums.ResultStatus2: ACLR for the first NR channel with lower frequency
- Carrier: enums.ResultStatus2: Power in the allocated NR channel
- Nr_Pos: enums.ResultStatus2: ACLR for the first NR 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 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
- Nr_Neg: float: ACLR for the first NR channel with lower frequency
- Carrier: float: Power in the allocated NR channel
- Nr_Pos: float: ACLR for the first NR 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(*sEGMent=SEGMent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:ACLR:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.current.
↪calculate(sEGMent = repcap.SEGMent.Default)
```

Return ACLR single value results for NR standalone mode, list mode segment <no>. 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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:ACLR:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.current.
↪fetch(sEGMent = repcap.SEGMent.Default)
```

Return ACLR single value results for NR standalone mode, list mode segment <no>. 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.

6.2.1.6.5.4 Dallocation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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 transmission bandwidth

fetch(*sEGMent=SEGMent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:ACLR:DALLocation
value: FetchStruct = driver.nrSubMeas.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.

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.

6.2.1.6.5.5 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:ACLR:DCHType
```

class DchTypeCls

DchType 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
- Channel_Type: enums.ChannelTypeA: No parameter help available

fetch(*sEGMent=SEGMent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:ACLR:DCHType
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.dchType.
↪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.

6.2.1.6.5.6 Endc

class EndcCls

Endc 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.nrSubMeas.multiEval.listPy.segment.aclr.endc.clone()
```

Subgroups

6.2.1.6.5.7 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:ENDC:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:ENDC: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
- Nr_Neg: enums.ResultStatus2: No parameter help available
- Carrier: enums.ResultStatus2: Power in the allocated channel (aggregated BW)
- Nr_Pos: enums.ResultStatus2: 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
- Nr_Neg: float: No parameter help available
- Carrier: float: Power in the allocated channel (aggregated BW)
- Nr_Pos: float: No parameter help available

calculate(*sEGMent=SEGMent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:ENDC:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.endc.
↪average.calculate(sEGMent = repcap.SEGMent.Default)
```

Returns ACLR single value results for EN-DC mode, list mode segment <no>. 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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:ENDC:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.endc.
↪average.fetch(sEGMent = repcap.SEGMent.Default)
```

Returns ACLR single value results for EN-DC mode, list mode segment <no>. 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.

6.2.1.6.5.8 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:ENDC:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:ACLR:ENDC: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
- Nr_Neg: enums.ResultStatus2: No parameter help available
- Carrier: enums.ResultStatus2: Power in the allocated channel (aggregated BW)
- Nr_Pos: enums.ResultStatus2: 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
- Nr_Neg: float: No parameter help available
- Carrier: float: Power in the allocated channel (aggregated BW)
- Nr_Pos: float: No parameter help available

calculate(*sEGMent=SEGMent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:ENDC:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.endc.
↪current.calculate(sEGMent = repcap.SEGMent.Default)
```

Returns ACLR single value results for EN-DC mode, list mode segment <no>. 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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:ACLR:ENDC:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.aclr.endc.
↪current.fetch(sEGMent = repcap.SEGMent.Default)
```

Returns ACLR single value results for EN-DC mode, list mode segment <no>. 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.

6.2.1.6.5.9 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.listPy.segment.cc.repcap_carrierComponent_get()
driver.nrSubMeas.multiEval.listPy.segment.cc.repcap_carrierComponent_set(repcap.
↪CarrierComponent.Nr1)
```

class CcCls

Cc commands group definition. 25 total commands, 3 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.nrSubMeas.multiEval.listPy.segment.cc.clone()
```

Subgroups

6.2.1.6.5.10 EsFlatness

class EsFlatnessCls

EsFlatness 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.nrSubMeas.multiEval.listPy.segment.cc.esFlatness.clone()
```

Subgroups

6.2.1.6.5.11 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:ESFLatness:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.
↪esFlatness.average.calculate(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode, for carrier <c>.

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 CalculateStruct structure arguments.

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.esFlatness.
↪average.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode, for carrier <c>.

Suppressed linked return values: reliability

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.

6.2.1.6.5.12 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:ESFLatness:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.
↪esFlatness.current.calculate(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode, for carrier <c>.

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 CalculateStruct structure arguments.

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.esFlatness.
↪current.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode, for carrier <c>.

Suppressed linked return values: reliability

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.nrSubMeas.multiEval.listPy.segment.cc.esFlatness.current.clone()
```

Subgroups

6.2.1.6.5.13 ScIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:CURRent:SCINdex
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.esFlatness.
↪current.scIndex.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return subcarrier indices of the equalizer spectrum flatness measurement for segment <no> in list mode, for carrier <c>. 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')

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.

6.2.1.6.5.14 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:ESFLatness:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:EXTReme
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.
```

(continues on next page)

(continued from previous page)

```
↪esFlatness.extreme.calculate(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode, for carrier <c>.

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 CalculateStruct structure arguments.

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:EXTReme
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.esFlatness.
↪extreme.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode, for carrier <c>.

Suppressed linked return values: reliability

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.

6.2.1.6.5.15 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:ESFLatness:SDEViation
```

class StandardDevCls

StandardDev commands group definition. 1 total commands, 0 Subgroups, 1 group commands

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)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:ESFLatness:SDEViation
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.esFlatness.
↪standardDev.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return equalizer spectrum flatness single value results for segment <no> in list mode, for carrier <c>.

Suppressed linked return values: reliability

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.

6.2.1.6.5.16 Iemission

class IemissionCls

Iemission commands group definition. 7 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.clone()
```

Subgroups

6.2.1.6.5.17 Margin

class MarginCls

Margin 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.nrSubMeas.multiEval.listPy.segment.cc.iemission.margin.clone()
```

Subgroups

6.2.1.6.5.18 Average

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:IEMission:MARGin:AVERage
```

class AverageCls

Average 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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:IEMission:MARGin:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.
↪margin.average.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for segment <no> in list mode, for carrier <c>. 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.nrSubMeas.multiEval.listPy.segment.cc.iemission.margin.average.clone()
```

Subgroups

6.2.1.6.5.19 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:IEMission:MARGin:AVERage: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 for the general margin (at non-allocated RBs)
- Iq_Image: int: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: int: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:IEMission:MARGin:AVERage:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.
↪margin.average.rbIndex.fetch(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the in-band emission measurement for segment <no> in list mode, for carrier <c>. At these RB indices, the CURRent, AVERage 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.

6.2.1.6.5.20 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:IEMission: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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:IEMission:MARGin:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.
↪margin.current.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for segment <no> in list mode, for carrier <c>. 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.nrSubMeas.multiEval.listPy.segment.cc.iemission.margin.current.clone()
```

Subgroups

6.2.1.6.5.21 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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: '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 for the general margin (at non-allocated RBs)
- Iq_Image: int: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: int: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:IEMission:MARGin:CURRent:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.
↪margin.current.rbIndex.fetch(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the in-band emission measurement for segment <no> in list mode, for carrier <c>. At these RB indices, the CURRent, AVERage 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.

6.2.1.6.5.22 Extreme

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:IEMission: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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:IEMission:MARGin:EXTReme
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.
↪margin.extreme.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for segment <no> in list mode, for carrier <c>. 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.nrSubMeas.multiEval.listPy.segment.cc.iemission.margin.extreme.clone()
```

Subgroups

6.2.1.6.5.23 RbIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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: '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 for the general margin (at non-allocated RBs)
- Iq_Image: int: Resource block index for the I/Q image margin (at image frequencies of allocated RBs)
- Carr_Leakage: int: Resource block index for the carrier leakage margin (at carrier frequency)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:IEMission:MARGin:EXTReme:RBINdex
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.
↪margin.extreme.rbIndex.fetch(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return resource block indices of the in-band emission measurement for segment <no> in list mode, for carrier <c>. At these RB indices, the CURRent, AVERage 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.

6.2.1.6.5.24 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:IEMission: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: Margin over all non-allocated RBs (scope of general limit component)
- Iq_Image: float: Margin at image frequencies of allocated RBs (scope of I/Q image limit component)
- Carr_Leakage: float: Margin at the carrier frequency (scope of I/Q offset limit component)

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:IEMission:MARGin:SDEViation
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.iemission.
↪margin.standardDev.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent =␣
↪repcap.CarrierComponent.Default)
```

Return the in-band emission limit line margin results for segment <no> in list mode, for carrier <c>. 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.

6.2.1.6.5.25 Modulation

class ModulationCls

Modulation commands group definition. 10 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.clone()
```

Subgroups

6.2.1.6.5.26 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:MODulation:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:MODulation: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
- 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*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:AVERage
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.
↪modulation.average.calculate(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Returns modulation single-value results for segment <no> in list mode, for carrier <c>. 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')

param carrierComponent
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*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.
↪average.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Returns modulation single-value results for segment <no> in list mode, for carrier <c>. 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')

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.

6.2.1.6.5.27 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:MODulation:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:CURRent
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.
↪modulation.current.calculate(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Returns modulation single-value results for segment <no> in list mode, for carrier <c>. 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')

param carrierComponent
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*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.
↪current.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Returns modulation single-value results for segment <no> in list mode, for carrier <c>. 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')

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.

6.2.1.6.5.28 Dallocation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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 transmission bandwidth

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:DALLocation
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.
↪dallocation.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return the detected allocation for segment <no> in list mode. The result is determined from the last measured slot of the statistical length.

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.

6.2.1.6.5.29 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:MODulation:DCHType
```

class DchTypeCls

DchType 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
- Channel_Type: enums.ChannelTypeA: No parameter help available

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:DCHType
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.
```

(continues on next page)

(continued from previous page)

```
↪dchType.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

No command help available

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.

6.2.1.6.5.30 Dmodulation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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: BPSK, BPWS: /2-BPSK, /2-BPSK with shaping QPSK, Q16, Q64, Q256: QPSK, 16QAM, 64QAM, 256QAM

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:DMODulation
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.
↪dmodulation.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Returns the detected modulation scheme for segment <no> in list mode, for carrier <c>. The result is determined from the last measured slot of the statistical length.

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.

6.2.1.6.5.31 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:MODulation:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]: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: No parameter help available
- Peak_Power_Min: float or bool: No parameter help available
- 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: No parameter help available
- Peak_Power_Min: float: No parameter help available
- 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*, *carrierComponent=CarrierComponent.Default*) → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:EXTReme
value: CalculateStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.
↪modulation.extreme.calculate(sEGMent = repcap.SEGMent.Default,␣
↪carrierComponent = repcap.CarrierComponent.Default)
```

Return modulation single-value results for segment <no> in list mode, for carrier <c>. 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')

param carrierComponent
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*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:EXTReme
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.
↪extreme.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Return modulation single-value results for segment <no> in list mode, for carrier <c>. 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')

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.

6.2.1.6.5.32 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC<Carrier>
↪]:MODulation: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
- 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

fetch(*sEGMent=SEGMent.Default*, *carrierComponent=CarrierComponent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>[:CC
↪<Carrier>]:MODulation:SDEViation
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.cc.modulation.
↪standardDev.fetch(sEGMent = repcap.SEGMent.Default, carrierComponent = repcap.
↪CarrierComponent.Default)
```

Returns modulation single-value results for segment <no> in list mode, for carrier <c>. 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')

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.

6.2.1.6.5.33 Pmonitor

class PmonitorCls

Pmonitor 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.nrSubMeas.multiEval.listPy.segment.pmonitor.clone()
```

Subgroups

6.2.1.6.5.34 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.nrSubMeas.multiEval.listPy.segment.pmonitor.array.clone()
```

Subgroups

6.2.1.6.5.35 Length

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:PMONitor:ARRay:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*sEGMent=SEGMent.Default*) → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:PMONitor:ARRay:LENGth
value: int = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.array.length.
↪fetch(sEGMent = repcap.SEGMent.Default)
```

Returns the number of power monitor vs subframe 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_NrFr1Meas.NrSubMeas.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 vs subframe results.

6.2.1.6.5.36 Start

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:PMONitor:ARRay:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*sEGMent=SEGMent.Default*) → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:PMONitor:ARRay:STARt
value: int = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.array.start.
↪fetch(sEGMent = repcap.SEGMent.Default)
```

Returns the offset of the first power monitor vs subframe 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_NrFr1Meas.NrSubMeas.MultiEval.ListPy. Pmonitor.Rms.fetch. A returned <Start> value n indicates that the result for the first subframe of the segment is the result (n+1) 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 vs subframe result.

6.2.1.6.5.37 Peak

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:PEAK
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.peak.
↪fetch(sEGMent = repcap.SEGMent.Default)
```

Return the power monitor vs subframe 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.

6.2.1.6.5.38 Rms

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:PMONitor:RMS
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.rms.
↪fetch(sEGMent = repcap.SEGMent.Default)
```

Return the power monitor vs subframe 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.

6.2.1.6.5.39 Slots

class SlotsCls

Slots 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.nrSubMeas.multiEval.listPy.segment.pmonitor.slots.clone()
```

Subgroups

6.2.1.6.5.40 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.nrSubMeas.multiEval.listPy.segment.pmonitor.slots.array.clone()
```

Subgroups

6.2.1.6.5.41 Length

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:SLOTs:ARRay:LENGth
```

class LengthCls

Length commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*sEGMent=SEGMent.Default*) → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:SLOTs:ARRay:LENGth
value: int = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.slots.array.
↪length.fetch(sEGMent = repcap.SEGMent.Default)
```

Returns the number of power monitor vs slot 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_NrFr1Meas.NrSubMeas.MultiEval.ListPy.Pmonitor. Slots.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 vs slot results.

6.2.1.6.5.42 Start

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:SLOTs:ARRay:STARt
```

class StartCls

Start commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*sEGMent=SEGMent.Default*) → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:SLOTs:ARRay:STARt
value: int = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.slots.array.
↪start.fetch(sEGMent = repcap.SEGMent.Default)
```

Returns the offset of the first power monitor vs slot 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_NrFr1Meas.NrSubMeas.MultiEval.ListPy. Pmonitor.Slots.Rms.fetch. A returned <Start> value n indicates that the result for the first slot of the segment is the result (n+1) 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 vs slot result.

6.2.1.6.5.43 Peak

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>:PMONitor:SLOTs: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 slot, from first to last slot of the segment For an inactive segment, only one INV is returned, independent of the number of slots in the segment.

fetch(*sEGMent=SEGMent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr>
↪:PMONitor:SLOTs:PEAK
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.slots.
↪peak.fetch(sEGMent = repcap.SEGMent.Default)
```

Return the power monitor vs slot results for segment <no> in list mode. The commands return one power result for each slot of the segment for the measured carrier. The power values are RMS averaged over the slot or represent the peak value within the slot.

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.

6.2.1.6.5.44 Rms

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:PMONitor:SLOTs: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 slot, from first to last slot of the segment For an inactive segment, only one INV is returned, independent of the number of slots in the segment.

fetch(*sEGMent=SEGMent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:PMONitor:SLOTs:RMS
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.pmonitor.slots.
↪rms.fetch(sEGMent = repcap.SEGMent.Default)
```

Return the power monitor vs slot results for segment <no> in list mode. The commands return one power result for each slot of the segment for the measured carrier. The power values are RMS averaged over the slot or represent the peak value within the slot.

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.

6.2.1.6.5.45 Power

class PowerCls

Power 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.nrSubMeas.multiEval.listPy.segment.power.clone()
```

Subgroups

6.2.1.6.5.46 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:POWer:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:AVERage
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:AVERage
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.47 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:POWer:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:CURRent
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:CURRent
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.48 Maximum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:POWer:MAXimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:MAXimum
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:MAXimum
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.49 Minimum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:POWer:MINimum
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:MINimum
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:MINimum
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.50 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:POWer:SDEViation
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.51 SeMask

class SeMaskCls

SeMask commands group definition. 16 total commands, 7 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.listPy.segment.seMask.clone()
```

Subgroups

6.2.1.6.5.52 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:SEMask:AVERage
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:AVERage
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:AVERage
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.53 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:SEMask:CURRent
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:CURRent
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:CURRent
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.54 Dallocation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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 transmission bandwidth

fetch(*sEGMent=SEGMent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:DALLocation
value: FetchStruct = driver.nrSubMeas.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.

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.

6.2.1.6.5.55 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:SEMask:DCHType
```

class DchTypeCls

DchType 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
- Channel_Type: enums.ChannelTypeA: No parameter help available

fetch(*sEGMent=SEGMent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:DCHType
value: FetchStruct = driver.nrSubMeas.multiEval.listPy.segment.seMask.dchType.
↪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.

6.2.1.6.5.56 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:SEMask:EXTReme
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:EXTReme
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:EXTReme
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.57 Margin

class MarginCls

Margin 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.nrSubMeas.multiEval.listPy.segment.seMask.margin.clone()
```

Subgroups

6.2.1.6.5.58 All

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:MARGin:ALL
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.59 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.nrSubMeas.multiEval.listPy.segment.seMask.margin.average.clone()
```

Subgroups

6.2.1.6.5.60 Negativ

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:MARGin:AVERage:NEGativ
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.61 Positiv

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:MARGin:AVERage:POSitiv
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.62 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.nrSubMeas.multiEval.listPy.segment.seMask.margin.current.clone()
```

Subgroups

6.2.1.6.5.63 Negativ

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:MARGin:CURRent:NEGativ
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.64 Positiv

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:MARGin:CURRent:POSitiv
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.65 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.nrSubMeas.multiEval.listPy.segment.seMask.margin.minimum.clone()
```

Subgroups

6.2.1.6.5.66 Negativ

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:MARGin:MINimum:NEGativ
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.67 Positiv

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:MARGin:MINimum:POSitiv
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.5.68 StandardDev

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>:SEMask: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
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot

fetch(*sEGMent=SEGMent.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEGMent<nr6g>
↪:SEMask:SDEViation
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.6 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.nrSubMeas.multiEval.listPy.seMask.clone()
```

Subgroups

6.2.1.6.6.1 Dallocation

SCPI Command :

```
FETCh:NRSub: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 transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:DALLocation
value: FetchStruct = driver.nrSubMeas.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 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.

6.2.1.6.6.2 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[ChannelTypeA]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:DCHType
value: List[enums.ChannelTypeA] = driver.nrSubMeas.multiEval.listPy.seMask.
↪dchType.fetch()
```

No command help available

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.6.6.3 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.nrSubMeas.multiEval.listPy.seMask.margin.clone()
```

Subgroups

6.2.1.6.6.4 Area<Area>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.nrSubMeas.multiEval.listPy.seMask.margin.area.repcap_area_get()
driver.nrSubMeas.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.nrSubMeas.multiEval.listPy.seMask.margin.area.clone()
```

Subgroups

6.2.1.6.6.5 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.nrSubMeas.multiEval.listPy.seMask.margin.area.negativ.clone()
```

Subgroups

6.2.1.6.6.6 Average

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA
↪<nr6g>:NEGativ:AVERage
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.6.7 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA
↪<nr6g>:NEGativ:CURRent
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.6.8 Minimum

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA
↪<nr6g>:NEGativ:MINimum
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.6.9 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.nrSubMeas.multiEval.listPy.seMask.margin.area.positiv.clone()
```

Subgroups

6.2.1.6.6.10 Average

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA
↪<nr6g>:POSitiv:AVERage
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.6.11 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA
↪<nr6g>:POSitiv:CURRent
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.6.12 Minimum

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA<nr6g>
↪: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MARGin:AREA
↪<nr6g>:POSitiv:MINimum
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.6.6.13 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.nrSubMeas.multiEval.listPy.seMask.obw.clone()
```

Subgroups

6.2.1.6.6.14 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:OBW:AVERage
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:AVERage
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.15 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:OBW:CURRent
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:CURRent
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.16 Extreme

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:OBW:EXTReme
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:EXTReme
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.17 StandardDev

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:OBW:SDEViation
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.18 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.nrSubMeas.multiEval.listPy.seMask.txPower.clone()
```

Subgroups

6.2.1.6.6.19 Average

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:AVERage
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:AVERage
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:AVERage
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.20 Current

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:CURRent
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:CURRent
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:CURRent
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.21 Maximum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MAXimum
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MAXimum
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MAXimum
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.22 Minimum

SCPI Commands :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:TXPower:MINimum
CALCulate:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MINimum
value: List[float or bool] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:MINimum
value: List[float] = driver.nrSubMeas.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

6.2.1.6.6.23 StandardDev

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:LIST:SEMask:TXPower:SDEViation
value: List[float] = driver.nrSubMeas.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

6.2.1.6.7 Sreliability

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SRELiability
```

class SreliabilityCls

Sreliability commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:LIST:SRELiability
value: List[int] = driver.nrSubMeas.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.

6.2.1.7 Pmonitor

class PmonitorCls

Pmonitor 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.nrSubMeas.multiEval.pmonitor.clone()
```

Subgroups

6.2.1.7.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:PMONitor: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:NRSub:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:AVERage
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.7.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:PMONitor: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:NRSub:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:CURRent
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.7.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:PMONitor: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:NRSub:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.7.4 Minimum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:MINimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:PMONitor: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:NRSub:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.minimum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:MINimum
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.minimum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.7.5 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
FETCh:NRSub:MEASurement<Instance>:MEValuation:PMONitor: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:NRSub:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.standardDev.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:PMONitor:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.pmonitor.standardDev.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.8 ReferenceMarker

class ReferenceMarkerCls

ReferenceMarker 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.nrSubMeas.multiEval.referenceMarker.clone()
```

Subgroups

6.2.1.8.1 Pdynamics

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:REFMarker:PDYNamics
value: float = driver.nrSubMeas.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
Absolute x-value of the marker position

param trace_select
No help available

return
yvalue: Absolute y-value of the marker position

6.2.1.8.2 Pmonitor

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:REFMarker:PMONitor
```

class PmonitorCls

Pmonitor commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch(*xvalue: int*) → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:REFMarker:PMONitor
value: float = driver.nrSubMeas.multiEval.referenceMarker.pmonitor.fetch(xvalue
↪= 1)
```

Uses the reference marker on the power monitor trace.

Suppressed linked return values: reliability

param xvalue
Absolute x-value of the marker position (slot number)

return
yvalue: Absolute y-value of the marker position

6.2.1.9 SeMask

class SeMaskCls

SeMask commands group definition. 22 total commands, 9 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.seMask.clone()
```

Subgroups

6.2.1.9.1 Alength

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:ALENgth
```

class AlengthCls

Alength commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:ALENgth
value: List[int] = driver.nrSubMeas.multiEval.seMask.alength.fetch()
```

No command help available

Suppressed linked return values: reliability

return
length: No help available

6.2.1.9.2 Aoffset

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:AOFFset
```

class AoffsetCls

Aoffset commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:AOFFset
value: List[int] = driver.nrSubMeas.multiEval.seMask.aoffset.fetch()
```

No command help available

Suppressed linked return values: reliability

return
offset: No help available

6.2.1.9.3 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:SEMask:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:AVERage
CALCulate:NRSub: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

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

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SEMask:AVERage
value: ResultData = driver.nrSubMeas.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.

6.2.1.9.4 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:SEMask:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:CURRent
CALCulate:NRSub: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

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

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SEMask:CURRent
value: ResultData = driver.nrSubMeas.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.

6.2.1.9.5 Dallocation

SCPI Command :

```
FETCh:NRSub: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 transmission bandwidth

fetch() → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:DALLocation
value: FetchStruct = driver.nrSubMeas.multiEval.seMask.dallocation.fetch()
```

Returns the detected allocation for the measured slot.

return
structure: for return value, see the help for FetchStruct structure arguments.

6.2.1.9.6 DchType

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:DCHType
```

class DchTypeCls

DchType commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → ChannelTypeA

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:DCHType
value: enums.ChannelTypeA = driver.nrSubMeas.multiEval.seMask.dchType.fetch()
```

No command help available

Suppressed linked return values: reliability

return
channel_type: No help available

6.2.1.9.7 Extreme

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:SEMask:EXTReme
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:EXTReme
CALCulate:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SEMask:EXTReme
value: ResultData = driver.nrSubMeas.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.

6.2.1.9.8 Margin

class MarginCls

Margin 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.nrSubMeas.multiEval.seMask.margin.clone()
```

Subgroups

6.2.1.9.8.1 All

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:ALL
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.9.8.2 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.nrSubMeas.multiEval.seMask.margin.average.clone()
```

Subgroups

6.2.1.9.8.3 Negativ

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:AVERage:NEGativ
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.9.8.4 Positiv

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:AVERage:POSitiv
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.9.8.5 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.nrSubMeas.multiEval.seMask.margin.current.clone()
```

Subgroups

6.2.1.9.8.6 Negativ

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:CURRent:NEGativ
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.9.8.7 Positiv

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:CURRent:POSitiv
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.9.8.8 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.nrSubMeas.multiEval.seMask.margin.minimum.clone()
```

Subgroups

6.2.1.9.8.9 Negativ

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:MINimum:NEGativ
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.9.8.10 Positiv

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>
↪:MEValuation:SEMask:MARGin:MINimum:POSitiv
value: FetchStruct = driver.nrSubMeas.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.

6.2.1.9.9 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:SEMask:SDEViation
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask: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 spectrum emission measurements exceeding the specified spectrum emission mask limits.
- Obw: float: Occupied bandwidth
- Tx_Power: float: Total TX power in the slot

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:SDEViation
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SEMask:SDEViation
value: ResultData = driver.nrSubMeas.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.

6.2.1.10 State

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:STATe
value: enums.ResourceState = driver.nrSubMeas.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.nrSubMeas.multiEval.state.clone()
```

Subgroups

6.2.1.10.1 All

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:MEValuation:STATe:ALL
value: List[enums.ResourceState] = driver.nrSubMeas.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

6.2.1.11 Trace

class TraceCls

Trace commands group definition. 52 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.clone()
```

Subgroups

6.2.1.11.1 Aclr

class AclrCls

Aclr commands group definition. 8 total commands, 3 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.aclr.clone()
```

Subgroups

6.2.1.11.1.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
FETCh:NRSub: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
- Nr_Neg: float: Power in the first NR channel with lower frequency
- Carrier: float: Power in the allocated NR channel
- Nr_Pos: float: Power in the first NR 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:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.average.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram for NR standalone. 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:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:AVERage
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.average.read()
```

Returns the absolute powers as displayed in the ACLR diagram for NR standalone. 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.

6.2.1.11.1.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
FETCh:NRSub: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
- Nr_Neg: float: Power in the first NR channel with lower frequency
- Carrier: float: Power in the allocated NR channel
- Nr_Pos: float: Power in the first NR 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:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.current.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram for NR standalone. 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:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:CURRent
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.current.read()
```

Returns the absolute powers as displayed in the ACLR diagram for NR standalone. 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.

6.2.1.11.1.3 Endc

class EndcCls

Endc 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.nrSubMeas.multiEval.trace.aclr.endc.clone()
```

Subgroups

6.2.1.11.1.4 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Endc_Neg: float: Power in the adjacent channel with lower frequency
- Carrier: float: Power in the allocated channel (aggregated BW)
- Endc_Pos: float: Power in the adjacent channel with higher frequency

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:AVERage
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.endc.average.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram for EN-DC. 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:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:AVERage
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.endc.average.read()
```

Returns the absolute powers as displayed in the ACLR diagram for EN-DC. 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.

6.2.1.11.1.5 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Endc_Neg: float: Power in the adjacent channel with lower frequency
- Carrier: float: Power in the allocated channel (aggregated BW)
- Endc_Pos: float: Power in the adjacent channel with higher frequency

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:CURRent
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.endc.current.fetch()
```

Returns the absolute powers as displayed in the ACLR diagram for EN-DC. 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:NRSub:MEASurement<Instance>:MEValuation:TRACe:ACLR:ENDC:CURRent
value: ResultData = driver.nrSubMeas.multiEval.trace.aclr.endc.current.read()
```

Returns the absolute powers as displayed in the ACLR diagram for EN-DC. 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.

6.2.1.11.2 Cc<CarrierComponent>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.trace.cc.repcap_carrierComponent_get()
driver.nrSubMeas.multiEval.trace.cc.repcap_carrierComponent_set(repcap.CarrierComponent.
↪Nr1)
```

class CcCls

Cc commands group definition. 20 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.nrSubMeas.multiEval.trace.cc.clone()
```

Subgroups

6.2.1.11.2.1 Layer<Layer>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.trace.cc.layer.repcap_layer_get()
driver.nrSubMeas.multiEval.trace.cc.layer.repcap_layer_set(repcap.Layer.Nr1)
```

class LayerCls

Layer commands group definition. 20 total commands, 5 Subgroups, 0 group commands Repeated Capability: Layer, default value after init: Layer.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.cc.layer.clone()
```

Subgroups

6.2.1.11.2.2 EsFlatness

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>]:ESFLatness
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>]:ESFLatness
```

class EsFlatnessCls

EsFlatness commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:ESFLatness
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.esFlatness.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the equalizer spectrum flatness trace for carrier <no>, layer/antenna <l>. See also 'Square Equalizer Spectrum Flatness'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:ESFLatness
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.esFlatness.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the equalizer spectrum flatness trace for carrier <no>, layer/antenna <l>. See also 'Square Equalizer Spectrum Flatness'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

6.2.1.11.2.3 Evmc

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>]:EVMC
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>]:EVMC
```

class EvmcCls

Evmc commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMC
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmc.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the EVM vs subcarrier trace for carrier <no>, layer/antenna <l>. See also 'Square EVM vs Subcarrier'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
ratio: Comma-separated list of EVM values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMC
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmc.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the EVM vs subcarrier trace for carrier <no>, layer/antenna <l>. See also 'Square EVM vs Subcarrier'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
ratio: Comma-separated list of EVM values, one value per subcarrier For not allocated subcarriers, NCAP is returned.

6.2.1.11.2.4 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.nrSubMeas.multiEval.trace.cc.layer.evmSymbol.clone()
```

Subgroups

6.2.1.11.2.5 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:EVMSymbol:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:EVMSymbol:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMSymbol:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmSymbol.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM vs modulation symbol trace for carrier <no>, layer/antenna <l>. See also ‘Square EVM’.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Cc’)

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface ‘Layer’)

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMSymbol:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmSymbol.
```

(continues on next page)

(continued from previous page)

```
↪average.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM vs modulation symbol trace for carrier <no>, layer/antenna <l>. See also 'Square EVM'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

6.2.1.11.2.6 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:EVMSymbol:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:EVMSymbol:CURRent
```

`class CurrentCls`

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMSymbol:CURRent
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmSymbol.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM vs modulation symbol trace for carrier <no>, layer/antenna <l>. See also 'Square EVM'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMSymbol:CURRent
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmSymbol.
↪current.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM vs modulation symbol trace for carrier <no>, layer/antenna <l>. See also 'Square EVM'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

6.2.1.11.2.7 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:EVMSymbol:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:EVMSymbol:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMSymbol:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmSymbol.
↪maximum.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM vs modulation symbol trace for carrier <no>, layer/antenna <l>. See also 'Square EVM'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:EVMSymbol:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.evmSymbol.
↪maximum.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the EVM vs modulation symbol trace for carrier <no>, layer/antenna <l>. See also 'Square EVM'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
ratio: Comma-separated list of EVM values, one value per modulation symbol

6.2.1.11.2.8 Iemissions

class IemissionsCls

Iemissions 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.nrSubMeas.multiEval.trace.cc.layer.iemissions.clone()
```

Subgroups

6.2.1.11.2.9 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.
↪average.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the in-band emissions trace for carrier <no>, layer/antenna <l>. The current, average and maximum traces can be retrieved. See also 'Square Inband Emission'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per resource block

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.
↪average.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the in-band emissions trace for carrier <no>, layer/antenna <l>. The current, average and maximum traces can be retrieved. See also 'Square Inband Emission'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per resource block

6.2.1.11.2.10 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions[:CURRent]
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions[:CURRent]
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions[:CURRent]
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.
↪current.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the in-band emissions trace for carrier <no>, layer/antenna <l>. The current, average and maximum traces can be retrieved. See also 'Square Inband Emission'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per resource block

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions[:CURRent]
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.
↪current.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the in-band emissions trace for carrier <no>, layer/antenna <l>. The current, average and maximum traces can be retrieved. See also 'Square Inband Emission'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per resource block

6.2.1.11.2.11 Limit

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions:LIMit
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions:LIMit
```

class LimitCls

Limit commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions:LIMit
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.limit.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the values of the in-band emissions limit line for carrier <no>, layer/antenna <l>.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
limit: Comma-separated list of limit values, one value per resource block

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions:LIMit
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.limit.
↪read(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.Layer.
↪Default)
```

Returns the values of the in-band emissions limit line for carrier <no>, layer/antenna <l>.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
limit: Comma-separated list of limit values, one value per resource block

6.2.1.11.2.12 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>
↪]:IEMissions:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.
↪maximum.fetch(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the in-band emissions trace for carrier <no>, layer/antenna <l>. The current, average and maximum traces can be retrieved. See also 'Square Inband Emission'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per resource block

read(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IEMissions:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.cc.layer.iemissions.
↪maximum.read(carrierComponent = repcap.CarrierComponent.Default, layer =␣
↪repcap.Layer.Default)
```

Returns the values of the in-band emissions trace for carrier <no>, layer/antenna <l>. The current, average and maximum traces can be retrieved. See also 'Square Inband Emission'.

Suppressed linked return values: reliability

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: Comma-separated list of power values, one value per resource block

6.2.1.11.2.13 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.nrSubMeas.multiEval.trace.cc.layer.iq.clone()
```

Subgroups

6.2.1.11.2.14 High

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IQ:HIGH
value: FetchStruct = driver.nrSubMeas.multiEval.trace.cc.layer.iq.high.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the results in the I/Q constellation diagram for low and high EVM window position, for carrier <no>, layer/antenna <l>. 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'.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.2.1.11.2.15 Low

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer<layer>]: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(*carrierComponent=CarrierComponent.Default*, *layer=Layer.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<no>][:LAYer
↪<layer>]:IQ:LOW
value: FetchStruct = driver.nrSubMeas.multiEval.trace.cc.layer.iq.low.
↪fetch(carrierComponent = repcap.CarrierComponent.Default, layer = repcap.
↪Layer.Default)
```

Returns the results in the I/Q constellation diagram for low and high EVM window position, for carrier <no>, layer/antenna <l>. 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'.

param carrierComponent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Cc')

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
structure: for return value, see the help for FetchStruct structure arguments.

6.2.1.11.3 Layer<Layer>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.nrSubMeas.multiEval.trace.layer.repcap_layer_get()
driver.nrSubMeas.multiEval.trace.layer.repcap_layer_set(repcap.Layer.Nr1)
```

class LayerCls

Layer commands group definition. 6 total commands, 1 Subgroups, 0 group commands Repeated Capability: Layer, default value after init: Layer.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.layer.clone()
```

Subgroups

6.2.1.11.3.1 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.nrSubMeas.multiEval.trace.layer.pdynamics.clone()
```

Subgroups

6.2.1.11.3.2 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>]:PDYNamics:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>]:PDYNamics:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>
↪]:PDYNamics:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.layer.pdynamics.average.
↪fetch(layer = repcap.Layer.Default)
```

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. The OFF power sections refer to antenna <l>. The ON power section refers to the sum of both antenna signals. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measured active slot. The values have a spacing of 1.5625 µs. The 705th value is at the start of the slot (0 µs) .

read(*layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>
↪]:PDYNamics:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.layer.pdynamics.average.
↪read(layer = repcap.Layer.Default)
```

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. The OFF power sections refer to antenna <l>. The ON power section refers to the sum of both antenna signals. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measured active slot. The values have a spacing of 1.5625 µs. The 705th value is at the start of the slot (0 µs) .

6.2.1.11.3.3 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>]:PDYNamics:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>]:PDYNamics:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>
↪]:PDYNamics:CURRent
value: List[float] = driver.nrSubMeas.multiEval.trace.layer.pdynamics.current.
↪fetch(layer = repcap.Layer.Default)
```

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. The OFF power sections refer to antenna <l>. The ON power section refers to the sum of both antenna signals. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measured active slot. The values have a spacing of 1.5625 µs. The 705th value is at the start of the slot (0 µs) .

read(*layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>
↪]:PDYNamics:CURRent
value: List[float] = driver.nrSubMeas.multiEval.trace.layer.pdynamics.current.
↪read(layer = repcap.Layer.Default)
```

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. The OFF power sections refer to antenna <l>. The ON power section refers to the sum of both antenna signals. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measured active slot. The values have a spacing of 1.5625 µs. The 705th value is at the start of the slot (0 µs) .

6.2.1.11.3.4 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>]:PDYNamics:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>]:PDYNamics:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*layer=Layer.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>
↪]:PDYNamics:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.layer.pdynamics.maximum.
↪fetch(layer = repcap.Layer.Default)
```

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. The OFF power sections refer to antenna <l>. The ON power section refers to the sum of both antenna signals. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measured active slot. The values have a spacing of 1.5625 µs. The 705th value is at the start of the slot (0 µs) .

read(*layer=Layer.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe[:LAYer<layer>
↪]:PDYNamics:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.layer.pdynamics.maximum.
↪read(layer = repcap.Layer.Default)
```

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. The OFF power sections refer to antenna <l>. The ON power section refers to the sum of both antenna signals. See also 'Square Power Dynamics'.

Suppressed linked return values: reliability

param layer
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Layer')

return
power: 2048 power values, from -1100 µs to +2098.4375 µs relative to the start of the measured active slot. The values have a spacing of 1.5625 µs. The 705th value is at the start of the slot (0 µs) .

6.2.1.11.4 Pmonitor

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor
```

class PmonitorCls

Pmonitor commands group definition. 4 total commands, 1 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor
value: List[float] = driver.nrSubMeas.multiEval.trace.pmonitor.fetch()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor
value: List[float] = driver.nrSubMeas.multiEval.trace.pmonitor.read()
```

No command help available

Suppressed linked return values: reliability

return
power: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.pmonitor.clone()
```

Subgroups

6.2.1.11.4.1 Slots

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SLOTs
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SLOTs
```

class SlotsCls

Slots commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SLOTs
value: List[float] = driver.nrSubMeas.multiEval.trace.pmonitor.slots.fetch()
```

Returns the power monitor results as power vs slot values (as in the result diagram) . The number of subframes per trace is configurable, see method RsCMPX_NrFr1Meas.Configure.NrSubMeas.MultiEval.nsubFrames. The number of slots per subframe depends on the SCS.

Suppressed linked return values: reliability

return
power: Comma-separated list of power values, one value per slot

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor:SLOTs
value: List[float] = driver.nrSubMeas.multiEval.trace.pmonitor.slots.read()
```

Returns the power monitor results as power vs slot values (as in the result diagram) . The number of subframes per trace is configurable, see method RsCMPX_NrFr1Meas.Configure.NrSubMeas.MultiEval.nsubFrames. The number of slots per subframe depends on the SCS.

Suppressed linked return values: reliability

return
power: Comma-separated list of power values, one value per slot

6.2.1.11.5 SeMask

class SeMaskCls

SeMask 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.nrSubMeas.multiEval.trace.seMask.clone()
```

Subgroups

6.2.1.11.5.1 Area<Area>

RepCap Settings

```
# Range: Nr1 .. Nr12
rc = driver.nrSubMeas.multiEval.trace.seMask.area.repcap_area_get()
driver.nrSubMeas.multiEval.trace.seMask.area.repcap_area_set(repcap.Area.Nr1)
```

class AreaCls

Area commands group definition. 8 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.nrSubMeas.multiEval.trace.seMask.area.clone()
```

Subgroups

6.2.1.11.5.2 Negative

class NegativeCls

Negative commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.seMask.area.negative.clone()
```

Subgroups

6.2.1.11.5.3 Average

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:NEGative:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW
- Power: List[float]: Comma-separated list of power results If the limit check is disabled for the area, INV is returned.

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:NEGative:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.negative.
↪average.fetch(area = repcap.Area.Default)
```

Returns the measured power values for a single spectrum emission mask area with enabled limit check. The results of the current, average and maximum traces can be retrieved. The area is located below (NEGative) or above (POSitive) the carrier center frequency. See also 'Square Spectrum Emission Mask'.

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.

6.2.1.11.5.4 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:NEGative:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW

- Power: List[float]: Comma-separated list of power results If the limit check is disabled for the area, INV is returned.

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:NEGative:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.negative.
↪current.fetch(area = repcap.Area.Default)
```

Returns the measured power values for a single spectrum emission mask area with enabled limit check. The results of the current, average and maximum traces can be retrieved. The area is located below (NEGative) or above (POSitive) the carrier center frequency. See also 'Square Spectrum Emission Mask'.

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.

6.2.1.11.5.5 Frequency

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:NEGative:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW
- Frequency: List[float]: Comma-separated list of frequency values

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:NEGative:FREQuency
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.negative.
↪frequency.fetch(area = repcap.Area.Default)
```

Returns the frequencies of a single spectrum emission mask area. The area is located below (NEGative) or above (POSitive) the carrier center frequency. At the returned frequencies, the power values are measured, see method RsCMPX_NrFr1Meas. NrSubMeas.MultiEval.Trace.SeMask.Area.Positive.Current.fetch.

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.

6.2.1.11.5.6 Maximum

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:NEGative:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW
- Power: List[float]: Comma-separated list of power results If the limit check is disabled for the area, INV is returned.

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:NEGative:MAXimum
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.negative.
↪maximum.fetch(area = repcap.Area.Default)
```

Returns the measured power values for a single spectrum emission mask area with enabled limit check. The results of the current, average and maximum traces can be retrieved. The area is located below (NEGative) or above (POSitive) the carrier center frequency. See also 'Square Spectrum Emission Mask'.

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.

6.2.1.11.5.7 Positive

class PositiveCls

Positive commands group definition. 4 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.seMask.area.positive.clone()
```

Subgroups

6.2.1.11.5.8 Average

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:POSitive:AVERage
```

class AverageCls

Average commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW
- Power: List[float]: Comma-separated list of power results If the limit check is disabled for the area, INV is returned.

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:POSitive:AVERage
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.positive.
↪average.fetch(area = repcap.Area.Default)
```

Returns the measured power values for a single spectrum emission mask area with enabled limit check. The results of the current, average and maximum traces can be retrieved. The area is located below (NEGative) or above (POSitive) the carrier center frequency. See also 'Square Spectrum Emission Mask'.

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.

6.2.1.11.5.9 Current

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:POSitive:CURRent
```

class CurrentCls

Current commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW

- Power: List[float]: Comma-separated list of power results If the limit check is disabled for the area, INV is returned.

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:POSitive:CURRent
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.positive.
↪current.fetch(area = repcap.Area.Default)
```

Returns the measured power values for a single spectrum emission mask area with enabled limit check. The results of the current, average and maximum traces can be retrieved. The area is located below (NEGative) or above (POSitive) the carrier center frequency. See also 'Square Spectrum Emission Mask'.

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.

6.2.1.11.5.10 Frequency

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:POSitive:FREQuency
```

class FrequencyCls

Frequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW
- Frequency: List[float]: Comma-separated list of frequency values

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:POSitive:FREQuency
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.positive.
↪frequency.fetch(area = repcap.Area.Default)
```

Returns the frequencies of a single spectrum emission mask area. The area is located below (NEGative) or above (POSitive) the carrier center frequency. At the returned frequencies, the power values are measured, see method RsCMPX_NrFr1Meas. NrSubMeas.MultiEval.Trace.SeMask.Area.Positive.Current.fetch.

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.

6.2.1.11.5.11 Maximum

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>:POSitive:MAXimum
```

class MaximumCls

Maximum commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: 'Reliability indicator'
- Rbw: enums.RbwB: Used resolution bandwidth (configured via limit settings) . K030: 30 kHz K100: 100 kHz K400: 400 kHz M1: 1 MHz PC1: 1 % of channel BW PC2: 2 % of channel BW
- Power: List[float]: Comma-separated list of power results If the limit check is disabled for the area, INV is returned.

fetch(*area=Area.Default*) → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:AREA<area>
↪:POSitive:MAXimum
value: FetchStruct = driver.nrSubMeas.multiEval.trace.seMask.area.positive.
↪maximum.fetch(area = repcap.Area.Default)
```

Returns the measured power values for a single spectrum emission mask area with enabled limit check. The results of the current, average and maximum traces can be retrieved. The area is located below (NEGative) or above (POSitive) the carrier center frequency. See also 'Square Spectrum Emission Mask'.

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.

6.2.1.11.5.12 Rbw<Rbw>

RepCap Settings

```
# Range: Bw1 .. Bw1000
rc = driver.nrSubMeas.multiEval.trace.seMask.rbw.repcap_rbw_get()
driver.nrSubMeas.multiEval.trace.seMask.rbw.repcap_rbw_set(repcap.Rbw.Bw1)
```

class RbwCls

Rbw commands group definition. 6 total commands, 3 Subgroups, 0 group commands Repeated Capability: Rbw, default value after init: Rbw.Bw1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.multiEval.trace.seMask.rbw.clone()
```

Subgroups

6.2.1.11.5.13 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rbw=Rbw.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.seMask.rbw.average.
↪fetch(rbw = repcap.Rbw.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 rbw
optional repeated capability selector. Default value: Bw1 (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. If there is no result for a frequency because no limit is defined, INV is returned.

read(*rbw=Rbw.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:AVERage
value: List[float] = driver.nrSubMeas.multiEval.trace.seMask.rbw.average.
↪read(rbw = repcap.Rbw.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 rbw
optional repeated capability selector. Default value: Bw1 (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. If there is no result for a frequency because no limit is defined, INV is returned.

6.2.1.11.5.14 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rbw=Rbw.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:CURRent
value: List[float] = driver.nrSubMeas.multiEval.trace.seMask.rbw.current.
↪fetch(rbw = repcap.Rbw.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 rbw
optional repeated capability selector. Default value: Bw1 (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. If there is no result for a frequency because no limit is defined, INV is returned.

read(*rbw=Rbw.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:CURRent
value: List[float] = driver.nrSubMeas.multiEval.trace.seMask.rbw.current.
↪read(rbw = repcap.Rbw.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 rbw
optional repeated capability selector. Default value: Bw1 (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. If there is no result for a frequency because no limit is defined, INV is returned.

6.2.1.11.5.15 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch(*rbw=Rbw.Default*) → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.seMask.rbw.maximum.
↪fetch(rbw = repcap.Rbw.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 rbw
optional repeated capability selector. Default value: Bw1 (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. If there is no result for a frequency because no limit is defined, INV is returned.

read(*rbw=Rbw.Default*) → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:RBW<kHz>
↪:MAXimum
value: List[float] = driver.nrSubMeas.multiEval.trace.seMask.rbw.maximum.
↪read(rbw = repcap.Rbw.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 rbw
optional repeated capability selector. Default value: Bw1 (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. If there is no result for a frequency because no limit is defined, INV is returned.

6.2.1.12 TxPower

class TxPowerCls

TxPower 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.nrSubMeas.multiEval.txPower.clone()
```

Subgroups

6.2.1.12.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:AVERage
FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower: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: For future use, currently no related limits
- Tx_Power: float: Total TX power

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower:AVERage
value: ResultData = driver.nrSubMeas.multiEval.txPower.average.fetch()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:AVERage
value: ResultData = driver.nrSubMeas.multiEval.txPower.average.read()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.12.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:CURRent
FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower: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: For future use, currently no related limits
- Tx_Power: float: Total TX power

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower:CURRent
value: ResultData = driver.nrSubMeas.multiEval.txPower.current.fetch()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:CURRent
value: ResultData = driver.nrSubMeas.multiEval.txPower.current.read()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.12.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:MAXimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower: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: For future use, currently no related limits
- Tx_Power: float: Total TX power

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.txPower.maximum.fetch()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:MAXimum
value: ResultData = driver.nrSubMeas.multiEval.txPower.maximum.read()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.12.4 Minimum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:MINimum
FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower: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: For future use, currently no related limits
- Tx_Power: float: Total TX power

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower:MINimum
value: ResultData = driver.nrSubMeas.multiEval.txPower.minimum.fetch()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:MINimum
value: ResultData = driver.nrSubMeas.multiEval.txPower.minimum.read()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.12.5 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:SDEViation
FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower: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: For future use, currently no related limits
- Tx_Power: float: Total TX power

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:TXPower:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.txPower.standardDev.fetch()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:MEValuation:TXPower:SDEViation
value: ResultData = driver.nrSubMeas.multiEval.txPower.standardDev.read()
```

Return the total TX power results (current, average, minimum, maximum and standard deviation) .

return
structure: for return value, see the help for ResultData structure arguments.

6.2.1.13 VfThroughput

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:MEValuation:VFTHroughput
```

class VfThroughputCls

VfThroughput commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:MEValuation:VFTHroughput
value: float = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh
STOP:NRSub:MEASurement<Instance>:PRACh
ABORt:NRSub:MEASurement<Instance>:PRACh
```

class PrachCls

Prach commands group definition. 83 total commands, 5 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:NRSub:MEASurement<Instance>:PRACh
driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh
driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh
driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh
driver.nrSubMeas.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.
    - 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␣
```

(continues on next page)

(continued from previous page)

```
↪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_NrFr1Meas.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.nrSubMeas.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.nrSubMeas.prach.evmSymbol.clone()
```

Subgroups

6.2.2.1.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
FETCh:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
CALCulate:NRSub: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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: CalculateStruct = driver.nrSubMeas.prach.evmSymbol.average.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. See also 'Square EVM vs Symbol'.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:AVERage
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. 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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
FETCh:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
CALCulate:NRSub: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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: CalculateStruct = driver.nrSubMeas.prach.evmSymbol.current.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. See also 'Square EVM vs Symbol'.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:CURRent
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. 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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
FETCh:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
CALCulate:NRSub: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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: CalculateStruct = driver.nrSubMeas.prach.evmSymbol.maximum.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. See also 'Square EVM vs Symbol'.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:MAXimum
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. 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.nrSubMeas.prach.evmSymbol.peak.clone()
```

Subgroups

6.2.2.1.4.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. See also 'Square EVM vs Symbol'.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:AVERage
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. 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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. See also 'Square EVM vs Symbol'.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:CURRent
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. 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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. See also 'Square EVM vs Symbol'.

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.nrSubMeas.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 11 If the preamble contains fewer than 12 symbols, NCAPs are returned for the remaining symbols. 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. 20 total commands, 9 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.prach.modulation.clone()
```

Subgroups

6.2.2.2.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:MODulation:AVERage
FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:AVERage
CALCulate:NRSub: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: Time error

- Tx_Power: float or bool: User equipment power
- Peak_Power: float or bool: User equipment 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: float: User equipment power
- Peak_Power: float: User equipment peak power

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:PRACh:MODulation:AVERage
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:AVERage
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:AVERage
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:CURRent
FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:CURRent
CALCulate:NRSub: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: Time error

- Tx_Power: float or bool: User equipment power
- Peak_Power: float or bool: User equipment 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: float: User equipment power
- Peak_Power: float: User equipment peak power

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:PRACh:MODulation:CURRent
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:CURRent
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:CURRent
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:DPFoffset
```

class DpfOffsetCls

DpfOffset commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch() → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:DPFoffset
value: int = driver.nrSubMeas.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.nrSubMeas.prach.modulation.dpfOffset.clone()
```

Subgroups

6.2.2.2.3.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr64
rc = driver.nrSubMeas.prach.modulation.dpfOffset.preamble.repcap_preamble_get()
driver.nrSubMeas.prach.modulation.dpfOffset.preamble.repcap_preamble_set(repcap.Preamble.
↪Nr1)
```

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:MODulation:DPFoffset:PREamble
↪<Number>
value: int = driver.nrSubMeas.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.nrSubMeas.prach.modulation.dpfOffset.preamble.clone()
```

6.2.2.2.4 DsIndex

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:DSINdex
```

class DsIndexCls

DsIndex commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch() → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:DSINdex
value: int = driver.nrSubMeas.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.nrSubMeas.prach.modulation.dsIndex.clone()
```

Subgroups

6.2.2.2.4.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr64
rc = driver.nrSubMeas.prach.modulation.dsIndex.preamble.repcap_preamble_get()
driver.nrSubMeas.prach.modulation.dsIndex.preamble.repcap_preamble_set(repcap.Preamble.
↪Nr1)
```

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:MODulation:DSINdex:PREamble
↪<Number>
value: int = driver.nrSubMeas.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.nrSubMeas.prach.modulation.dsIndex.preamble.clone()
```

6.2.2.2.5 Extreme

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:MODulation:EXTReme
FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:EXTReme
CALCulate:NRSub: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 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

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 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

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:EXTReme
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MODulation:NSYMbol
```

class NsymbolCls

Nsymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:NSYMbol
value: int = driver.nrSubMeas.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 .. Nr64
rc = driver.nrSubMeas.prach.modulation.preamble.repcap_preamble_get()
driver.nrSubMeas.prach.modulation.preamble.repcap_preamble_set(repcap.Preamble.Nr1)
```

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
FETCh:NRSub: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: Time error
- Tx_Power: float: User equipment power
- Peak_Power: float: User equipment peak power

fetch(*preamble=Preamble.Default*) → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
value: ResultData = driver.nrSubMeas.prach.modulation.preamble.fetch(preamble =␣
↪repcap.Preamble.Default)
```

Return the single value results of the EVM vs Preamble and Power vs Preamble squares, for a selected preamble. See also 'Squares 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:NRSub:MEASurement<Instance>:PRACh:MODulation:PREamble<Number>
value: ResultData = driver.nrSubMeas.prach.modulation.preamble.read(preamble =␣
↪repcap.Preamble.Default)
```

Return the single value results of the EVM vs Preamble and Power vs Preamble squares, for a selected preamble. See also 'Squares 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.nrSubMeas.prach.modulation.preamble.clone()
```

6.2.2.2.8 Scorrelation

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:SCORrelation
```

class ScorrelationCls

Scorrelation commands group definition. 2 total commands, 1 Subgroups, 1 group commands

fetch() → float

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:SCORrelation
value: float = driver.nrSubMeas.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.nrSubMeas.prach.modulation.scorrelation.clone()
```

Subgroups

6.2.2.2.8.1 Preamble<Preamble>

RepCap Settings

```
# Range: Nr1 .. Nr64
rc = driver.nrSubMeas.prach.modulation.scorrelation.preamble.repcap_preamble_get()
driver.nrSubMeas.prach.modulation.scorrelation.preamble.repcap_preamble_set(repcap.
↪Preamble.Nr1)
```

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:MODulation:SCORrelation:PREamble
↪<Number>
value: float = driver.nrSubMeas.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.nrSubMeas.prach.modulation.scorrelation.preamble.clone()
```

6.2.2.2.9 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:MODulation:SDEViation
FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation: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 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: float: User equipment power
- Peak_Power: float: User equipment peak power

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:MODulation:SDEViation
value: ResultData = driver.nrSubMeas.prach.modulation.standardDev.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:MODulation:SDEViation
value: ResultData = driver.nrSubMeas.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. 14 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.prach.pdynamics.clone()
```

Subgroups

6.2.2.3.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:PDYNamics:AVERage
FETCh:NRSub:MEASurement<Instance>:PRACh:PDYNamics:AVERage
CALCulate:NRSub: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 before the preamble, without transient period
- On_Power_Rms: float or bool: ON power mean value over the preamble.
- On_Power_Peak: float or bool: ON power peak value within the preamble
- Off_Power_After: float or bool: OFF power after the 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 before the preamble, without transient period
- On_Power_Rms: float: ON power mean value over the preamble.
- On_Power_Peak: float: ON power peak value within the preamble
- Off_Power_After: float: OFF power after the preamble, without transient period

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:AVERage
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:CURRent
FETCh:NRSub:MEASurement<Instance>:PRACh:PDYNamics:CURRent
CALCulate:NRSub: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 before the preamble, without transient period

- On_Power_Rms: float or bool: ON power mean value over the preamble.
- On_Power_Peak: float or bool: ON power peak value within the preamble
- Off_Power_After: float or bool: OFF power after the 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 before the preamble, without transient period
- On_Power_Rms: float: ON power mean value over the preamble.
- On_Power_Peak: float: ON power peak value within the preamble
- Off_Power_After: float: OFF power after the preamble, without transient period

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:CURRent
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
FETCh:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
CALCulate:NRSub: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 before the preamble, without transient period
- On_Power_Rms: float or bool: ON power mean value over the preamble.
- On_Power_Peak: float or bool: ON power peak value within the preamble
- Off_Power_After: float or bool: OFF power after the 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 before the preamble, without transient period
- On_Power_Rms: float: ON power mean value over the preamble.
- On_Power_Peak: float: ON power peak value within the preamble
- Off_Power_After: float: OFF power after the preamble, without transient period

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MAXimum
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MINimum
FETCh:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MINimum
CALCulate:NRSub: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 before the preamble, without transient period
- On_Power_Rms: float or bool: ON power mean value over the preamble.
- On_Power_Peak: float or bool: ON power peak value within the preamble
- Off_Power_After: float or bool: OFF power after the 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 before the preamble, without transient period
- On_Power_Rms: float: ON power mean value over the preamble.
- On_Power_Peak: float: ON power peak value within the preamble

- Off_Power_After: float: OFF power after the preamble, without transient period

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:MINimum
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
FETCh:NRSub:MEASurement<Instance>:PRACh:PDYNamics: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 dynamics measurements exceeding the specified power dynamics limits.
- Off_Power_Before: float: OFF power before the preamble, without transient period

- On_Power_Rms: float: ON power mean value over the preamble.
- On_Power_Peak: float: ON power peak value within the preamble
- Off_Power_After: float: OFF power after the preamble, without transient period

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
value: ResultData = driver.nrSubMeas.prach.pdynamics.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:NRSub:MEASurement<Instance>:PRACh:PDYNamics:SDEViation
value: ResultData = driver.nrSubMeas.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:NRSub: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:NRSub:MEASurement<Instance>:PRACh:STATe
value: enums.ResourceState = driver.nrSubMeas.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.nrSubMeas.prach.state.clone()
```

Subgroups

6.2.2.4.1 All

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:STATe:ALL
value: List[enums.ResourceState] = driver.nrSubMeas.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.nrSubMeas.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.nrSubMeas.prach.trace.evm.clone()
```

Subgroups

6.2.2.5.1.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:AVERage
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, one value per subcarrier.

6.2.2.5.1.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:CURRent
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, one value per subcarrier.

6.2.2.5.1.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVM:MAXimum
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, one value per subcarrier.

6.2.2.5.2 EvPreamble

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVPReamble
FETCh:NRSub:MEASurement<Instance>:PRACh:TRACe:EVPReamble
```

class EvPreambleCls

EvPreamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:TRACe:EVPReamble
value: List[float] = driver.nrSubMeas.prach.trace.evPreamble.fetch()
```

Return the values of the EVM vs preamble traces. See also 'Squares EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, for preamble 1 to n of the measurement interval.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:EVPReamble
value: List[float] = driver.nrSubMeas.prach.trace.evPreamble.read()
```

Return the values of the EVM vs preamble traces. See also 'Squares EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: Comma-separated list of EVM values, for preamble 1 to n of the measurement interval.

6.2.2.5.3 Iq

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:IQ
value: FetchStruct = driver.nrSubMeas.prach.trace.iq.fetch()
```

Returns the results in the I/Q constellation diagram. There is one pair of values per modulation symbol. The number of modulation symbols equals the number of subcarriers and depends on the preamble format. 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.nrSubMeas.prach.trace.merror.clone()
```

Subgroups

6.2.2.5.4.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of magnitude error values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:AVERage
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of magnitude error values, one value per subcarrier.

6.2.2.5.4.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of magnitude error values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:CURRent
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of magnitude error values, one value per subcarrier.

6.2.2.5.4.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of magnitude error values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:MERRor:MAXimum
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of magnitude error values, one value per subcarrier.

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.nrSubMeas.prach.trace.pdynamics.clone()
```

Subgroups

6.2.2.5.5.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
value: List[float] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:AVERage
value: List[float] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
value: List[float] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:CURRent
value: List[float] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
value: List[float] = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TRACe:PDYNamics:MAXimum
value: List[float] = driver.nrSubMeas.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.nrSubMeas.prach.trace.perror.clone()
```

Subgroups

6.2.2.5.6.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of phase error values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:AVERage
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of phase error values, one value per subcarrier.

6.2.2.5.6.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of phase error values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:CURRent
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of phase error values, one value per subcarrier.

6.2.2.5.6.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
FETCh:NRSub: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:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of phase error values, one value per subcarrier.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PERRor:MAXimum
value: List[float] = driver.nrSubMeas.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 'Squares EVM, Magnitude Error, Phase Error'.

Suppressed linked return values: reliability

return
results: Comma-separated list of phase error values, one value per subcarrier.

6.2.2.5.7 PvPreamble

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PVPReamble
FETCh:NRSub:MEASurement<Instance>:PRACh:TRACe:PVPReamble
```

class PvPreambleCls

PvPreamble commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:PRACh:TRACe:PVPReamble
value: List[float] = driver.nrSubMeas.prach.trace.pvPreamble.fetch()
```

Return the values of the power vs preamble traces. See also 'Squares EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: Comma-separated list of power values, for preamble 1 to n of the measurement interval.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:PRACh:TRACe:PVPReamble
value: List[float] = driver.nrSubMeas.prach.trace.pvPreamble.read()
```

Return the values of the power vs preamble traces. See also 'Squares EVM vs Preamble, Power vs Preamble'.

Suppressed linked return values: reliability

return
results: Comma-separated list of power values, for preamble 1 to n of the measurement interval.

6.2.3 Srs

SCPI Commands :

```
INITiate:NRSub:MEASurement<Instance>:SRS
STOP:NRSub:MEASurement<Instance>:SRS
ABORt:NRSub:MEASurement<Instance>:SRS
```

class SrsCls

Srs commands group definition. 73 total commands, 6 Subgroups, 3 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:NRSub:MEASurement<Instance>:SRS
driver.nrSubMeas.srs.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:NRSub:MEASurement<Instance>:SRS
driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS
driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS
driver.nrSubMeas.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_NrFr1Meas.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.nrSubMeas.srs.clone()
```

Subgroups

6.2.3.1 EvmSymbol

class EvmSymbolCls

EvmSymbol 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.nrSubMeas.srs.evmSymbol.clone()
```

Subgroups

6.2.3.1.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:AVERage
FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol: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
- Low: List[float]: No parameter help available
- High: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:AVERage
value: ResultData = driver.nrSubMeas.srs.evmSymbol.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:AVERage
value: ResultData = driver.nrSubMeas.srs.evmSymbol.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.1.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:CURRent
FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol: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
- Low: List[float]: No parameter help available
- High: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:CURRent
value: ResultData = driver.nrSubMeas.srs.evmSymbol.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:CURRent
value: ResultData = driver.nrSubMeas.srs.evmSymbol.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.1.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:MAXimum
FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol: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
- Low: List[float]: No parameter help available
- High: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:MAXimum
value: ResultData = driver.nrSubMeas.srs.evmSymbol.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:MAXimum
value: ResultData = driver.nrSubMeas.srs.evmSymbol.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.1.4 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.nrSubMeas.srs.evmSymbol.peak.clone()
```

Subgroups

6.2.3.1.4.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:AVERage
FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK: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
- Low: List[float]: No parameter help available
- High: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:AVERage
value: ResultData = driver.nrSubMeas.srs.evmSymbol.peak.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:AVERage
value: ResultData = driver.nrSubMeas.srs.evmSymbol.peak.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.1.4.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:CURRent
FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK: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
- Low: List[float]: No parameter help available
- High: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:CURRent
value: ResultData = driver.nrSubMeas.srs.evmSymbol.peak.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:CURRent
value: ResultData = driver.nrSubMeas.srs.evmSymbol.peak.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.1.4.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:MAXimum
FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK: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
- Low: List[float]: No parameter help available
- High: List[float]: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.nrSubMeas.srs.evmSymbol.peak.maximum.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:EVMSymbol:PEAK:MAXimum
value: ResultData = driver.nrSubMeas.srs.evmSymbol.peak.maximum.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.2 Modulation

class ModulationCls

Modulation commands group definition. 12 total commands, 5 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.srs.modulation.clone()
```

Subgroups

6.2.3.2.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:MODulation:AVERage
FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:AVERage
CALCulate:NRSub:MEASurement<Instance>:SRS:MODulation: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
- 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: No parameter help available
- Out_Of_Tolerance: 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
- Frequency_Error: float: No parameter help available
- Timing_Error: float: No parameter help available
- Tx_Power: float: No parameter help available
- Peak_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:MODulation:AVERage
value: CalculateStruct = driver.nrSubMeas.srs.modulation.average.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:AVERage
value: ResultData = driver.nrSubMeas.srs.modulation.average.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:MODulation:AVERage
value: ResultData = driver.nrSubMeas.srs.modulation.average.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.2.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:MODulation:CURRent
FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:CURRent
CALCulate:NRSub:MEASurement<Instance>:SRS:MODulation: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
- 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: No parameter help available
- Out_Of_Tolerance: 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
- Frequency_Error: float: No parameter help available
- Timing_Error: float: No parameter help available
- Tx_Power: float: No parameter help available
- Peak_Power: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:MODulation:CURRent
value: CalculateStruct = driver.nrSubMeas.srs.modulation.current.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:CURRent
value: ResultData = driver.nrSubMeas.srs.modulation.current.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:MODulation:CURRent
value: ResultData = driver.nrSubMeas.srs.modulation.current.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.2.3 Extreme

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:MODulation:EXTReme
FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:EXTReme
CALCulate:NRSub:MEASurement<Instance>:SRS:MODulation:EXTReme
```

class ExtremeCls

Extreme 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_Minimum: float or bool: No parameter help available
- Tx_Power_Maximum: float or bool: No parameter help available
- Peak_Power_Min: float or bool: No parameter help available
- Peak_Power_Max: 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
- 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
- Frequency_Error: float: No parameter help available
- Timing_Error: float: No parameter help available
- Tx_Power_Minimum: float: No parameter help available
- Tx_Power_Maximum: float: No parameter help available
- Peak_Power_Min: float: No parameter help available
- Peak_Power_Max: float: No parameter help available

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:MODulation:EXTReme
value: CalculateStruct = driver.nrSubMeas.srs.modulation.extreme.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:EXTReme
value: ResultData = driver.nrSubMeas.srs.modulation.extreme.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:MODulation:EXTReme
value: ResultData = driver.nrSubMeas.srs.modulation.extreme.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.2.4 Nsymbol

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:NSYMbol
```

class NsymbolCls

Nsymbol commands group definition. 1 total commands, 0 Subgroups, 1 group commands

fetch() → int

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:NSYMbol
value: int = driver.nrSubMeas.srs.modulation.nsymbol.fetch()
```

No command help available

Suppressed linked return values: reliability

return
no_of_symbols: No help available

6.2.3.2.5 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:MODulation:SDEViation
FETCh:NRSub:MEASurement<Instance>:SRS:MODulation: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
- 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
- Frequency_Error: float: No parameter help available
- Timing_Error: float: No parameter help available
- Tx_Power: float: No parameter help available
- Peak_Power: float: No parameter help available

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:MODulation:SDEViation
value: ResultData = driver.nrSubMeas.srs.modulation.standardDev.fetch()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:MODulation:SDEViation
value: ResultData = driver.nrSubMeas.srs.modulation.standardDev.read()
```

No command help available

return
structure: for return value, see the help for ResultData structure arguments.

6.2.3.3 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.nrSubMeas.srs.pdynamics.clone()
```

Subgroups

6.2.3.3.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:PDYNamics:AVERage
FETCh:NRSub:MEASurement<Instance>:SRS:PDYNamics:AVERage
CALCulate:NRSub: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 SRS transmission.
- On_Power_Rms: float or bool: ON power mean value over the SRS transmission.
- On_Power_Peak: float or bool: ON power peak value for the SRS transmission
- Off_Power_After: float or bool: OFF power mean value for slot after SRS transmission.

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 SRS transmission.
- On_Power_Rms: float: ON power mean value over the SRS transmission.
- On_Power_Peak: float: ON power peak value for the SRS transmission
- Off_Power_After: float: OFF power mean value for slot after SRS transmission.

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:AVERage
value: ResultData = driver.nrSubMeas.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.

6.2.3.3.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:PDYNamics:CURRent
FETCh:NRSub:MEASurement<Instance>:SRS:PDYNamics:CURRent
CALCulate:NRSub: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 SRS transmission.
- On_Power_Rms: float or bool: ON power mean value over the SRS transmission.
- On_Power_Peak: float or bool: ON power peak value for the SRS transmission
- Off_Power_After: float or bool: OFF power mean value for slot after SRS transmission.

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 SRS transmission.
- On_Power_Rms: float: ON power mean value over the SRS transmission.
- On_Power_Peak: float: ON power peak value for the SRS transmission
- Off_Power_After: float: OFF power mean value for slot after SRS transmission.

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:CURRent
value: ResultData = driver.nrSubMeas.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.

6.2.3.3.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:PDYNamics:MAXimum
FETCh:NRSub:MEASurement<Instance>:SRS:PDYNamics:MAXimum
CALCulate:NRSub: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 SRS transmission.
- On_Power_Rms: float or bool: ON power mean value over the SRS transmission.
- On_Power_Peak: float or bool: ON power peak value for the SRS transmission
- Off_Power_After: float or bool: OFF power mean value for slot after SRS transmission.

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 SRS transmission.
- On_Power_Rms: float: ON power mean value over the SRS transmission.
- On_Power_Peak: float: ON power peak value for the SRS transmission
- Off_Power_After: float: OFF power mean value for slot after SRS transmission.

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:MAXimum
value: ResultData = driver.nrSubMeas.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.

6.2.3.3.4 Minimum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:PDYNamics:MINimum
FETCh:NRSub:MEASurement<Instance>:SRS:PDYNamics:MINimum
CALCulate:NRSub: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 SRS transmission.
- On_Power_Rms: float or bool: ON power mean value over the SRS transmission.
- On_Power_Peak: float or bool: ON power peak value for the SRS transmission
- Off_Power_After: float or bool: OFF power mean value for slot after SRS transmission.

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 SRS transmission.
- On_Power_Rms: float: ON power mean value over the SRS transmission.
- On_Power_Peak: float: ON power peak value for the SRS transmission
- Off_Power_After: float: OFF power mean value for slot after SRS transmission.

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: CalculateStruct = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:MINimum
value: ResultData = driver.nrSubMeas.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.

6.2.3.3.5 StandardDev

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:PDYNamics:SDEViation
FETCh:NRSub:MEASurement<Instance>:SRS:PDYNamics:SDEViation
CALCulate:NRSub: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: 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 the time period before SRS transmission.
- On_Power_Rms: float: ON power mean value over the SRS transmission.
- On_Power_Peak: float: ON power peak value for the SRS transmission
- Off_Power_After: float: OFF power mean value for slot after SRS transmission.

calculate() → CalculateStruct

```
# SCPI: CALCulate:NRSub:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: CalculateStruct = driver.nrSubMeas.srs.pdynamics.standardDev.calculate()
```

No command help available

return
structure: for return value, see the help for CalculateStruct structure arguments.

fetch() → ResultData

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: ResultData = driver.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:PDYNamics:SDEViation
value: ResultData = driver.nrSubMeas.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.

6.2.3.4 PvSymbol

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:PVSYmbol
FETCh:NRSub:MEASurement<Instance>:SRS:PVSYmbol
```

class PvSymbolCls

PvSymbol commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:PVSYmbol
value: List[float] = driver.nrSubMeas.srs.pvSymbol.fetch()
```

Return the single-value results of the power vs symbol measurement. See also 'Square Power vs Symbol'.

Suppressed linked return values: reliability

return
power: Comma-separated list of 10 average power values, one value per SRS transmission.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:PVSYmbol
value: List[float] = driver.nrSubMeas.srs.pvSymbol.read()
```

Return the single-value results of the power vs symbol measurement. See also 'Square Power vs Symbol'.

Suppressed linked return values: reliability

return
power: Comma-separated list of 10 average power values, one value per SRS transmission.

6.2.3.5 State

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:SRS:STATe
value: enums.ResourceState = driver.nrSubMeas.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.nrSubMeas.srs.state.clone()
```

Subgroups

6.2.3.5.1 All

SCPI Command :

```
FETCh:NRSub: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:NRSub:MEASurement<Instance>:SRS:STATe:ALL
value: List[enums.ResourceState] = driver.nrSubMeas.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

6.2.3.6 Trace

class TraceCls

Trace commands group definition. 27 total commands, 6 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.nrSubMeas.srs.trace.clone()
```

Subgroups

6.2.3.6.1 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.nrSubMeas.srs.trace.evm.clone()
```

Subgroups

6.2.3.6.1.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:AVERage
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:AVERage
value: List[float] = driver.nrSubMeas.srs.trace.evm.average.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:AVERage
value: List[float] = driver.nrSubMeas.srs.trace.evm.average.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.1.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:CURRent
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:CURRent
value: List[float] = driver.nrSubMeas.srs.trace.evm.current.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:CURRent
value: List[float] = driver.nrSubMeas.srs.trace.evm.current.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.1.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:MAXimum
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:MAXimum
value: List[float] = driver.nrSubMeas.srs.trace.evm.maximum.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:EVM:MAXimum
value: List[float] = driver.nrSubMeas.srs.trace.evm.maximum.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.2 Iq

SCPI Command :

```
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:IQ
```

class IqCls

Iq commands group definition. 1 total commands, 0 Subgroups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Iphase: List[float]: No parameter help available
- Qphase: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:IQ
value: FetchStruct = driver.nrSubMeas.srs.trace.iq.fetch()
```

No command help available

return
structure: for return value, see the help for FetchStruct structure arguments.

6.2.3.6.3 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.nrSubMeas.srs.trace.merror.clone()
```

Subgroups

6.2.3.6.3.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:AVERage
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:AVERage
value: List[float] = driver.nrSubMeas.srs.trace.merror.average.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:AVERage
value: List[float] = driver.nrSubMeas.srs.trace.merror.average.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.3.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:CURRent
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:CURRent
value: List[float] = driver.nrSubMeas.srs.trace.merror.current.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:CURRent
value: List[float] = driver.nrSubMeas.srs.trace.merror.current.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.3.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:MAXimum
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:MAXimum
value: List[float] = driver.nrSubMeas.srs.trace.merror.maximum.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:MERRor:MAXimum
value: List[float] = driver.nrSubMeas.srs.trace.merror.maximum.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.4 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.nrSubMeas.srs.trace.pdynamics.clone()
```

Subgroups

6.2.3.6.4.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
FETCh:NRSub: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:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
value: List[float] = driver.nrSubMeas.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. 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 SRS transmission. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS transmission (0 µs) .

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:AVERage
value: List[float] = driver.nrSubMeas.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. 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 SRS transmission. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS transmission (0 µs) .

6.2.3.6.4.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
FETCh:NRSub: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:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
value: List[float] = driver.nrSubMeas.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. 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 SRS transmission. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS transmission (0 µs) .

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:CURRent
value: List[float] = driver.nrSubMeas.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. 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 SRS transmission. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS transmission (0 µs) .

6.2.3.6.4.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
FETCh:NRSub: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:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
value: List[float] = driver.nrSubMeas.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. 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 SRS transmission. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS transmission (0 µs) .

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:PDYNamics:MAXimum
value: List[float] = driver.nrSubMeas.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. 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 SRS transmission. The values have a spacing of 1.5625 µs. The 705th value is at the start of the SRS transmission (0 µs) .

6.2.3.6.5 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.nrSubMeas.srs.trace.perror.clone()
```

Subgroups

6.2.3.6.5.1 Average

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:AVERage
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:AVERage
```

class AverageCls

Average commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:AVERage
value: List[float] = driver.nrSubMeas.srs.trace.perror.average.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:AVERage
value: List[float] = driver.nrSubMeas.srs.trace.perror.average.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.5.2 Current

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:CURRent
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:CURRent
```

class CurrentCls

Current commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:CURRent
value: List[float] = driver.nrSubMeas.srs.trace.perror.current.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:CURRent
value: List[float] = driver.nrSubMeas.srs.trace.perror.current.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.5.3 Maximum

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:MAXimum
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:MAXimum
```

class MaximumCls

Maximum commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:MAXimum
value: List[float] = driver.nrSubMeas.srs.trace.perror.maximum.fetch()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:PERRor:MAXimum
value: List[float] = driver.nrSubMeas.srs.trace.perror.maximum.read()
```

No command help available

Suppressed linked return values: reliability

return
results: No help available

6.2.3.6.6 PvSymbol

SCPI Commands :

```
READ:NRSub:MEASurement<Instance>:SRS:TRACe:PVSYmbol
FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PVSYmbol
```

class PvSymbolCls

PvSymbol commands group definition. 2 total commands, 0 Subgroups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:NRSub:MEASurement<Instance>:SRS:TRACe:PVSYmbol
value: List[float] = driver.nrSubMeas.srs.trace.pvSymbol.fetch()
```

Return the values of the power vs symbol trace. See also 'Square Power vs Symbol'.

Suppressed linked return values: reliability

return
power: Comma-separated list of power values, one value per symbol. The number of symbols depends on the SCS and on the configured number of subframes.

read() → List[float]

```
# SCPI: READ:NRSub:MEASurement<Instance>:SRS:TRACe:PVSYmbol
value: List[float] = driver.nrSubMeas.srs.trace.pvSymbol.read()
```

Return the values of the power vs symbol trace. See also 'Square Power vs Symbol'.

Suppressed linked return values: reliability

return
power: Comma-separated list of power values, one value per symbol. The number of symbols depends on the SCS and on the configured number of subframes.

6.3 Sense

class SenseCls

Sense 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.clone()
```

Subgroups

6.3.1 NrSubMeas

class NrSubMeasCls

NrSubMeas 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.nrSubMeas.clone()
```

Subgroups

6.3.1.1 ListPy

class ListPyCls

ListPy 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.nrSubMeas.listPy.clone()
```

Subgroups

6.3.1.1.1 Segment

RepCap Settings

```
# Range: Nr1 .. Nr512
rc = driver.sense.nrSubMeas.listPy.segment.repcap_sEGMent_get()
driver.sense.nrSubMeas.listPy.segment.repcap_sEGMent_set(repcap.SEGMent.Nr1)
```

class SegmentCls

Segment commands group definition. 1 total commands, 1 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.sense.nrSubMeas.listPy.segment.clone()
```

Subgroups

6.3.1.1.1.1 Cfrequency

SCPI Command :

```
SENSe:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:CFRequency
```

class CfrequencyCls

Cfrequency commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get(*sEGMent=SEGMent.Default*) → float

```
# SCPI: SENSe:NRSub:MEASurement<Instance>:LIST:SEGMent<no>:CFRequency
value: float = driver.sense.nrSubMeas.listPy.segment.cfrequency.get(sEGMent =␣
↪repcap.SEGMent.Default)
```

No command help available

param sEGMent
optional repeated capability selector. Default value: Nr1 (settable in the interface 'Segment')

return
frequency: No help available

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 NrSubMeas

class NrSubMeasCls

NrSubMeas commands group definition. 17 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrSubMeas.clone()
```

Subgroups

6.4.1.1 ListPy

SCPI Commands :

```
TRIGger:NRSub:MEASurement<Instance>:LIST:MODE
TRIGger:NRSub:MEASurement<Instance>:LIST:NBANdwidth
```

class ListPyCls

ListPy commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_mode() → ListMode

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:LIST:MODE
value: enums.ListMode = driver.trigger.nrSubMeas.listPy.get_mode()
```

Specifies the trigger mode for list mode measurements. For configuration of retrigger flags, see method RsCMPX_NrFr1Meas. Configure.NrSubMeas.ListPy.Segment.Setup.set. For configuration of the global trigger source, see TRIGger:NRSub: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() → NbTrigger

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:LIST:NBANdwidth
value: enums.NbTrigger = driver.trigger.nrSubMeas.listPy.get_nbandwidth()
```

Selects the trigger evaluation bandwidth for the retrigger source IFPNarrowband. Select the retrigger source via method RsCMPX_NrFr1Meas.Configure.NrSubMeas.ListPy.Segment.Setup.set.

return
nbandwidth: Evaluation bandwidth 10 MHz to 80 MHz

set_mode(*mode: ListMode*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:LIST:MODE
driver.trigger.nrSubMeas.listPy.set_mode(mode = enums.ListMode.ONCE)
```

Specifies the trigger mode for list mode measurements. For configuration of retrigger flags, see method RsCMPX_NrFr1Meas. Configure.NrSubMeas.ListPy.Segment.Setup.set. For configuration of the global trigger source, see TRIGger:NRSub: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(*nbandwidth: NbTrigger*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:LIST:NBANdwidth
driver.trigger.nrSubMeas.listPy.set_nbandwidth(nbandwidth = enums.NbTrigger.
↪M010)
```

Selects the trigger evaluation bandwidth for the retrigger source IFPNarrowband. Select the retrigger source via method RsCMPX_NrFr1Meas.Configure.NrSubMeas.ListPy.Segment.Setup.set.

param nbandwidth
Evaluation bandwidth 10 MHz to 80 MHz

6.4.1.2 MultiEval

SCPI Commands :

```
TRIGger:NRSub:MEASurement<Instance>:MEValuation:THReshold
TRIGger:NRSub:MEASurement<Instance>:MEValuation:SLOPe
TRIGger:NRSub:MEASurement<Instance>:MEValuation:DELay
TRIGger:NRSub:MEASurement<Instance>:MEValuation:TOUT
TRIGger:NRSub:MEASurement<Instance>:MEValuation:MGAP
TRIGger:NRSub:MEASurement<Instance>:MEValuation:SMODe
TRIGger:NRSub:MEASurement<Instance>:MEValuation:FSYNc
```

class MultiEvalCls

MultiEval commands group definition. 7 total commands, 0 Subgroups, 7 group commands

get_delay() → float

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:DELay
value: float = driver.trigger.nrSubMeas.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_fsync() → bool

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:FSYNc
value: bool = driver.trigger.nrSubMeas.multiEval.get_fsync()
```

Enables frame synchronization for 'Free Run (Fast Sync) ' and 'IF Power'.

return
frame_sync: OFF: slot synchronization ON: frame synchronization

get_mgap() → float

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:MGAP
value: float = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SLOPe
value: enums.SignalSlope = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SMODe
value: enums.SyncMode = driver.trigger.nrSubMeas.multiEval.get_smode()
```

Selects the size of the search window for synchronization.

return
sync_mode: Normal, enhanced, normal single slot, enhanced single slot

get_threshold() → float

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:THReshold
value: float or bool = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:TOUT
value: float or bool = driver.trigger.nrSubMeas.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_delay(*delay: float*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:DELay
driver.trigger.nrSubMeas.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_fsync(*frame_sync: bool*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:FSYNc
driver.trigger.nrSubMeas.multiEval.set_fsync(frame_sync = False)
```

Enables frame synchronization for 'Free Run (Fast Sync) ' and 'IF Power'.

param frame_sync
OFF: slot synchronization ON: frame synchronization

set_mgap(*min_trig_gap: float*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:MGAP
driver.trigger.nrSubMeas.multiEval.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:NRSub:MEASurement<Instance>:MEValuation:SLOPe
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:SMODe
driver.trigger.nrSubMeas.multiEval.set_smode(sync_mode = enums.SyncMode.
↪ENHanced)
```

Selects the size of the search window for synchronization.

param sync_mode
Normal, enhanced, normal single slot, enhanced single slot

set_threshold(*trig_threshold: float*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:THReshold
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:MEValuation:TOUT
driver.trigger.nrSubMeas.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

6.4.1.3 Prach

SCPI Commands :

```
TRIGger:NRSub:MEASurement<Instance>:PRACh:THReshold
TRIGger:NRSub:MEASurement<Instance>:PRACh:SLOPe
TRIGger:NRSub:MEASurement<Instance>:PRACh:TOUT
TRIGger:NRSub:MEASurement<Instance>:PRACh:MGAP
```

class PrachCls

Prach commands group definition. 4 total commands, 0 Subgroups, 4 group commands

get_mgap() → float

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:PRACh:MGAP
value: float = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:SLOPe
value: enums.SignalSlope = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:THReshold
value: float or bool = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TOUT
value: float or bool = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:MGAP
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:SLOPe
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:THReshold
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:PRACh:TOUT
driver.trigger.nrSubMeas.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

6.4.1.4 Srs

SCPI Commands :

```
TRIGger:NRSub:MEASurement<Instance>:SRS:THReshold
TRIGger:NRSub:MEASurement<Instance>:SRS:SLOPe
TRIGger:NRSub:MEASurement<Instance>:SRS:TOUT
TRIGger:NRSub:MEASurement<Instance>:SRS:MGAP
```

class SrsCls

Srs commands group definition. 4 total commands, 0 Subgroups, 4 group commands

get_mgap() → float

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:SRS:MGAP
value: float = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:SLOPe
value: enums.SignalSlope = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:THReshold
value: float or bool = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:TOUT
value: float or bool = driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:MGAP
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:SLOPe
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:THReshold
driver.trigger.nrSubMeas.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:NRSub:MEASurement<Instance>:SRS:TOUT
driver.trigger.nrSubMeas.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_NRFR1MEAS UTILITIES

class Utilities

Common utility class. Utility functions common for all types of drivers.

Access snippet: `utils = RsCMPX_NrFr1Meas.utilities`

property logger: *ScpiLogger*

Scpi Logger interface, see *here*

Access snippet: `logger = RsCMPX_NrFr1Meas.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_NrFr1Meas instance with new VISA session, the session gets a new thread lock. You can assign it to other RsCMPX_NrFr1Meas sessions in order to share one physical instrument with a multi-thread access.
- If you create new RsCMPX_NrFr1Meas 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_NRFR1MEAS 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_NRFR1MEAS 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:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MEASurement:ACLR:ENDC:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MEASurement:ESFLatness:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MEASurement:ESFLatness:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MEASurement:ESFLatness:EXTReme?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MEASurement:MODulation:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MEASurement:MODulation:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MEASurement:MODulation:EXTReme?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MOBAndwidth:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MOBAndwidth:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MOBAndwidth:EXTReme?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MXPSweep:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MXPSweep:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MXPSweep:MAXimum?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SEMask:MXPSweep:MINimum?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:DIFFerence:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:DIFFerence:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:DIFFerence:EXTReme?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:MAXPeak:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:MAXPeak:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:MAXPeak:EXTReme?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:MINPeak:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:MINPeak:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:MINPeak:EXTReme?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:RIPPle:AVERage?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:RIPPle:CURRent?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:RIPPle:EXTReme?
CALCulate:NRSub:MEASurement<Instance>:MEValuation:LIST:SPECtrum:MODulation:EVM:DMRS:HIGH:AVERage?
```

```
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SE
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIMit:SE
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST,
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:CMC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:CMW
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:LRA
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:NCC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:OSI
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:LIST:SEC
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MAPType,
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MMODe,
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulati
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulati
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulati
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulati
CONFigure:NRSub:MEASurement<Instance>:MEValuation:MODulati
```

D

E

F

FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement<Instance>:MEValuation:TRACe:SEMask
604
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:OFFSet:MEASurement<Instance>:MEValuation:TRACe:SEMask
605
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:RANGe:MEASurement<Instance>:MEValuation:TRACe:SEMask
606
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:SUBRange:MEASurement<Instance>:MEValuation:TRACe:SEMask
608
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:SUBType:MEASurement<Instance>:MEValuation:TRACe:SEMask
608
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:TXBRange:MEASurement<Instance>:MEValuation:TRACe:SEMask
609
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement<Instance>:MEValuation:TRACe[:CC<nc
610
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement:NEGative<Instance>:MEValuation:TRACe[:CC<nc
611
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement:POSitive<Instance>:MEValuation:TRACe[:CC<nc
612
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement:NEGative:CURRent<Instance>:MEValuation:TRACe[:CC<nc
613
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement:POSitive:CURRent<Instance>:MEValuation:TRACe[:CC<nc
614
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement:NEGative:AVERage<Instance>:MEValuation:TRACe[:CC<nc
615
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MEASurement:POSitive:AVERage<Instance>:MEValuation:TRACe[:CC<nc
615
FETCh:NRSub:MEASurement<Instance>:MEValuation:SEMask:MARGin:MODify:MEASurement<Instance>:MEValuation:TRACe[:CC<nc
616
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:NRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<nc
617
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:MRSub:MEASurement<Instance>:MEValuation:TRACe[:CC<nc
618
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:MRSub:MEASurement:AVERage<Instance>:MEValuation:TRACe[:CC<nc
619
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:MRSub:MEASurement:CURRent<Instance>:MEValuation:TRACe[:LAYer
620
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:MRSub:MEASurement:MAXimum:AVERage<Instance>:MEValuation:TRACe[:LAYer
621
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:MRSub:MEASurement:MAXimum:CURRent<Instance>:MEValuation:TRACe[:LAYer
622
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor:MEASurement<Instance>:MEValuation:TXPower:AVER
639
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PMONitor:MEASurement:TOFFset<Instance>:MEValuation:TXPower:CURR
640
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:MEASurement<Instance>:MEValuation:TXPower:MAXi
642
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:MEASurement:NEGative<Instance>:MEValuation:TXPower:MINi
642
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:MEASurement:NEGative:FREQuency<Instance>:MEValuation:TXPower:SDEV
643
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:SEMask:MEASurement:NEGative:MARGin<Instance>:MEValuation:VFTHroughput
644
645
645
646
647
648
649
650
623
625
626
627
628
629
631
632
630
634
635
636
637
638
651
652
653
653
654
655

FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
365
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
366
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
366
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
306
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
307
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
308
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
308
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
310
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
311
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
312
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
312
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
314
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
316
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
318
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
319
FETCh:NRSub:MEASurement<Instance>:MEValuation[:CC<no>][:LA
321
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer
322
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer
324
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer
326
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer
328
FETCh:NRSub:MEASurement<Instance>:MEValuation[:LAYer<layer
329
FETCh:NRSub:MEASurement<Instance>:MEValuation:AVERage,
330
FETCh:NRSub:MEASurement<Instance>:MEValuation:CURRent,
332
FETCh:NRSub:MEASurement<Instance>:MEValuation:MAXimum,
333
FETCh:NRSub:MEASurement<Instance>:MEValuation:PEAK:AVE
334
FETCh:NRSub:MEASurement<Instance>:MEValuation:PEAK:CUR
335
FETCh:NRSub:MEASurement<Instance>:MEValuation:PEAK:MAX
336

FETCh:NRSub:MEASurement<Instance>:MEValuation:MODulation:AVERage,
337
FETCh:NRSub:MEASurement<Instance>:MEValuation:MODulation:CURRent,
338
FETCh:NRSub:MEASurement<Instance>:MEValuation:MODulation:MAXimum,
339
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:AVERage,
341
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:CURRent,
342
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:MAXimum,
344
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:PEAK,
347
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:RESult,
351
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:RESult:CURRent,
354
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:RESult:MAXimum,
356
FETCh:NRSub:MEASurement<Instance>:MEValuation:SPECtrum:RESult:PEAK,
358
FETCh:NRSub:MEASurement<Instance>:MEValuation:SSTatistics:AVERage,
360
FETCh:NRSub:MEASurement<Instance>:MEValuation:SSTatistics:CURRent,
362
FETCh:NRSub:MEASurement<Instance>:MEValuation:SSTatistics:CURRent:SCINdex,
363
FETCh:NRSub:MEASurement<Instance>:MEValuation:SSTatistics:MAXimum,
363
FETCh:NRSub:MEASurement<Instance>:MEValuation:SSTatistics:PEAK,
364
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:AVERage,
369
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:CURRent,
371
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:MAXimum,
372
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PEAK:AVERage,
374
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PEAK:CURRent,
375
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRACe:PEAK:MAXimum,
657
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRANsient:PRACh:AVERage,
658
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRANsient:PRACh:CURRent,
660
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRANsient:PRACh:MAXimum,
661
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRANsient:PRACh:PEAK,
662
FETCh:NRSub:MEASurement<Instance>:MEValuation:TRANsient:PRACh:RESult,
663

G

I

L

M

R

U