Keysight M8000 Series of BER Test Solutions

J-BERT M8020A High-Performance BERT
M8030A Multi-Channel BERT
M8040A High-Performance BERT
Notices

© Keysight Technologies 2019
No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies as governed by United States and international copyright laws.

Trademarks
PCI Express® and PCIe® are registered trademarks of PCI-SIG.

Manual Part Number
M8000-91809

Edition
Edition 3.0, September 2019
Keysight Technologies Deutschland GmbH
Herrenberger Strasse 130,
71034 Böblingen, Germany

Technology Licenses
The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

U.S. Government Rights
The Software is “commercial computer software,” as defined by Federal Acquisition Regulation (“FAR”) 2.101. Pursuant to FAR 12.211 and 12.212, and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at http://www.keysight.com/find/sweula.

The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of those rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software.

With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFARS 227.7103-5 (c), as applicable in any technical data.

Safety Notices

CAUTION
A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING
A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED “AS IS,” AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.
Safety Summary

The following general safety precautions must be observed during all phases of operation of this instrument. Failure to comply with these precautions or with specific warnings or operating instructions in the product manuals violates safety standards of design, manufacture and intended use of the instrument. Keysight Technologies assumes no liability for the customer’s failure to comply with these requirements. Product manuals are provided with your instrument on CD-ROM and/or in printed form. Printed manuals are an option for many products. Manuals may also be available on the Web. Go to www.keysight.com and type in your product number in the Search field at the top of the page.

General
This product is a Safety Class 1 instrument (provided with a protective earth terminal). The protective features of this product may be impaired if it is used in a manner not specified in the operation instructions.

All Light Emitting Diodes (LEDs) used in this product are Class 1 LEDs as per IEC 60825-1.

Environment Conditions
This instrument is intended for indoor use in an installation category II, pollution degree 2 environment. It is designed to operate at a maximum relative humidity of 95% and at altitudes of up to 2000 meters.

Refer to the specifications tables for the ac mains voltage requirements and ambient operating temperature range.

Before Applying Power
Verify that all safety precautions are taken. The power cable inlet of the instrument serves as a device to disconnect from the mains in case of hazard. The instrument must be positioned so that the operator can easily access the power cable inlet. When the instrument is rack mounted the rack must be provided with an easily accessible mains switch.

Ground the Instrument
To minimize shock hazard, the instrument chassis and cover must be connected to an electrical protective earth ground. The instrument must be connected to the ac power mains through a grounded power cable, with the ground wire firmly connected to an electrical ground (safety ground) at the power outlet. Any interruption of the protective (grounding) conductor or disconnection of the protective earth terminal will cause a potential shock hazard that could result in personal injury.

Do Not Operate in an Explosive Atmosphere
Do not operate the instrument in the presence of flammable gases or fumes.

Do Not Remove the Instrument Cover
Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made only by qualified personnel.

Instruments that appear damaged or defective should be made inoperative and secured against unintended operation until they can be repaired by qualified service personnel.
## Safety Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Warning Symbol" /></td>
<td>Indicates warning or caution. If you see this symbol on a product, you must refer to the manuals for specific Warning or Caution information to avoid personal injury or damage to the product.</td>
</tr>
<tr>
<td><img src="image" alt="Frame Ground Symbol" /></td>
<td>Frame or chassis ground terminal. Typically connects to the equipment’s metal frame.</td>
</tr>
<tr>
<td><img src="image" alt="KC Symbol" /></td>
<td>KC is the Korean certification mark to demonstrate that the equipment is Class A suitable for professional use and is for use in electromagnetic environments outside of the home.</td>
</tr>
<tr>
<td><img src="image" alt="Static Protection Symbol" /></td>
<td>Indicates that antistatic precautions should be taken.</td>
</tr>
<tr>
<td><img src="image" alt="Time Period Symbol" /></td>
<td>Indicates the time period during which no hazardous or toxic substance elements are expected to leak or deteriorate during normal use. Forty years is the expected useful life of the product.</td>
</tr>
<tr>
<td><img src="image" alt="RCM Mark" /></td>
<td>The RCM Mark is a compliance mark to the ACMA (Australian Spectrum Management Agency). This indicates compliance with all Australian EMC regulatory information.</td>
</tr>
</tbody>
</table>
Compliance and Environmental Information

<table>
<thead>
<tr>
<th>Safety Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Safety Symbol" /></td>
<td>This product complies with WEEE Directive (2002/96/EC) marking requirements. The affixed label indicates that you must not discard this electrical/electronic product in domestic household waste.</td>
</tr>
<tr>
<td></td>
<td>Product Category: With reference to the equipment types in WEEE Directive Annex I, this product is classed as a “Monitoring and Control instrumentation” product.</td>
</tr>
<tr>
<td></td>
<td>Do not dispose in domestic household waste.</td>
</tr>
<tr>
<td></td>
<td>To return unwanted products, contact your local Keysight office, or see <a href="http://about.keysight.com/en/companyinfo/environment/takeback.shtml">http://about.keysight.com/en/companyinfo/environment/takeback.shtml</a> for more information.</td>
</tr>
</tbody>
</table>
Contents

Safety Summary 3
Safety Symbols 4
Compliance and Environmental Information 5

1 Programming Basics

Modular Configuration 14
M8020A Modular Configuration 14
M8030A Modular Configuration 17
M8040A Modular Configuration 19

Using Identifiers 21
Location Identifiers 21
Group Name Identifiers 22
Measurement Name Identifiers 22

PLUGin Subsystem 23

2 Recommended Programming Techniques

Output Protection 26
Pattern Generator Output Termination 26
Output Protection Algorithm 26

Controlling the Output Levels 27
Controlling the Output Levels - Concepts 27
Controlling the Output Levels - Procedures 27

Reading the M8020A/M8030A/M8040A’s Status 28
Reading the M8020A/M8030A/M8040A’s Status - Concepts 28
M8020A/M8030A/M8040A Status Reporting Structure 31
Program Snippet Extending the *opc? Functionality 35

Working with Patterns 36
Symbol Coding Descriptions 37
Symbol Sequences 42

Creating Pattern Sequences 43
Sequence Editor 43
Pattern Sequence Building Blocks 45
Sequencer Triggers 49
Block Branching 50
Link Training Events 51
Block Controls 51
Coding Configuration 52
Link Training Configuration 58
Downloading Pattern Sequence to Hardware 60

3 Programming Examples

Introduction 62

Initializing the Connection 62
N4903B Initialization 62
M8020A Initialization 63

SJ Example 64
N4903B SJ Example 64
M8020A SJ Example 66

PJ Example 70
N4903B PJ Example 70
M8020A PJ Example 72

Sampling Point Alignment Example 76
N4903B Sampling Point Alignment Example 76
M8020A Sampling Point Alignment Example 78
Contents

BER Example 80
  N4903B BER Example 80
  M8020A BER Example 81

M8070B Jitter Tolerance Measurement CSV Output 85

4 SCPI Command Language

SCPI Command Language - Introduction 88
  SCPI Common Commands 88
  SCPI Instrument Control Commands 88
  IEEE 488.2 Mandatory Commands 89
  Overlapped and Sequential Commands 89
  Data Types 90

Important Points about SCPI 91
  Important Points about SCPI - Concepts 91

Sending Commands to the M8020A/M8030A/M8040A 95
  Sending Commands to the M8020A/M8030A/M8040A - Overview 95
  Sending Commands using VISA 95

Command Line Arguments 96

Communication 97

5 SCPI Command Reference

Subsystems 100

IEEE Commands - Reference 101
  Mandatory Commands 101

SCPI Standard Commands 107

Miscellaneous Commands 108

Command Syntax to Find Min/Max Values 109
Contents

Location and Module Mapping - Reference 110

STATus Subsystem 111
  :STATus:QUESTionable Subnode 112
  :STATus:OPERation Subnode 121
  :STATus:INSTrument Subnode 125
  :STATus:INSTrument:ALIGnment Subnode 130

TRIGger Subsystem 132

Source Subsystem 139
  [:SOURce]:PULSe Subnode 144
  [:SOURce]:VOLTage Subnode 148
  [:SOURce]:JITTer Subnode 153
  [:SOURce]:JITTer:HFRequency Subnode 156
  [:SOURce]:JITTer:HFRequency:PERiodic Subnode 159
  [:SOURce]:JITTer:HFRequency:BUncorrelate Subnode 162
  [:SOURce]:JITTer:HFRequency:RANDom Subnode 166
  [:SOURce]:JITTer:HFRequency:SPECTrally Subnode 170
  [:SOURce]:JITTer:LFRequency Subnode 173
  [:SOURce]:JITTer:LFRequency:PERiodic Subnode 175
  [:SOURce]:JITTer:LFRequency:RSSClock Subnode 179
  [:SOURce]:JITTer:SWEep Subnode 181
  [:SOURce]:SSCLocking Subnode 187
  [:SOURce]:INTerference Subnode 193
  [:SOURce]:INTerference:RANDom Subnode 203
  [:SOURce]:INTerference:SINUsoidal Subnode 206
  [:SOURce]:CONFigure Subnode 210

OUTPut Subsystem 211
  :OUTPut:DEEMphasis Subnode 220
  :OUTPut:EINSertion Subnode 233
  :OUTPut:TRIGger Subnode 237
  :OUTPut:DEMBedding Subnode 239
  :OUTPut:EMBedding Subnode 242
  :OUTPut:ADJustment Subnode 245
<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT Subsystem</td>
<td>248</td>
</tr>
<tr>
<td>INPut Subsystem</td>
<td>250</td>
</tr>
<tr>
<td>SENSe Subsystem</td>
<td>298</td>
</tr>
<tr>
<td>CLOCK Subsystem</td>
<td>299</td>
</tr>
<tr>
<td>N1000 Subsystem</td>
<td>303</td>
</tr>
<tr>
<td>FETCH Subsystem</td>
<td>312</td>
</tr>
<tr>
<td>MMEMory Subsystem</td>
<td>317</td>
</tr>
<tr>
<td>ACQuisition sub</td>
<td>323</td>
</tr>
<tr>
<td>PLUGin Subsystem</td>
<td>328</td>
</tr>
<tr>
<td>SYSTem Subsystem</td>
<td>372</td>
</tr>
</tbody>
</table>
Contents

DATA Subsystem  391
   :DATA:PATTern Subnode  392
   :DATA:SEQuence Subnode  400
   :DATA:SYNC Subnode  409
   :DATA:LINecoding Subnode  411
   :DATA:FEC Subnode  415

Index
1 Programming Basics

- Modular Configuration / 14
- Using Identifiers / 21
- PLUGin Subsystem / 23
Modular Configuration

In order to describe remote programming for the M8020A, M8030A and M8040A, the modular configuration must be discussed.

The M8020A, M8030A and M8040A are modular instruments. They can consist of several numbers of components. A component can be an AXIe chassis, modules, channels, etc. All these components 'form' an instrument. These instruments can be controlled by M8070B system software.

The M8020A instrument supports the following modules.
- M8041A high-performance BERT generator-analyzer-clock 8/16 Gb/s
- M8051A high-performance BERT generator-analyzer 8/16 Gb/s
- M8061A multiplexer 2:1 with de-emphasis 32 Gb/s
- M8062A 32Gb/s Front-end for J-BERT M8020A High-Performance BERT

The M8020A modules must be installed in the M9505A 5-slot chassis.

The M8030A instrument supports the following modules.
- M8041A high-performance BERT generator-analyzer-clock 8/16 Gb/s
- M8051A high-performance BERT generator-analyzer 8/16 Gb/s
- M8192A Multi-channel synchronization module

The M8030A modules must be installed in the M9514A AXIe 14-slot chassis.

For complete details on the features and hardware components of each of the above mentioned modules, refer to M8020A and M8030A Getting Started Guide.

M8020A Modular Configuration

The following section describes and illustrates various setup combinations in which you can install the M8020A modules.

The following configurations are possible in an M9505A 5-slot chassis:
- 1 or 2-channel, 16 Gb/s - (1) M8041A
- 3 or 4-channel, 16 Gb/s - (1) M8041A + (1) M8051A
- 1-channel, 32 Gb/s (Pattern Generator only) - (1) M8041A + (1) M8061A
- 1-channel, 32 Gb/s (Pattern Generator only or full BERT) - (1) M8041A + (1) M8062A
1 or 2-channel System Configuration

The 1 or 2-channel configuration is a single channel system (a second channel can be added with license) consisting of the 5-slot M9505A AXIe Chassis and an M8041A module. The M8041A occupies three slots. A maximum of two M8020A modules can be installed in a 5-slot chassis.

3 or 4-channel System Configuration

The four channel configuration consists of the 5-slot M9505A AXIe Chassis, an M8041A module, and an M8051A module. The M8041A occupies three slots and the M8051A occupies two slots.

NOTE

This configuration requires a cable (provided with the M8051A) that connects the M8041A SYNC OUT to the M8051A SYNC IN to synchronize the two modules to a common system clock.
1-channel, 32 Gb/s (Pattern Generator Only)

A typical configuration using the M8061A 32 Gb/s multiplexer with de-emphasis consists of the 5-slot M9505A AXle Chassis, an M8041A module, and an M8061A module. The M8041A occupies three slots and the M8061A occupies two slots.
1-channel, 32 Gb/s (Pattern Generator only or full BERT)

A typical configuration for an M8020A 32 Gb/s full BERT consists of the 5-slot M9505A AXIe Chassis, an M8041A module, and an M8062A module. The M8041A occupies three slots and the M8062A occupies two slots.

![M8020A configuration for 32 Gb/s BERT](image)

**M8030A Modular Configuration**

The M8030A is a modular test solution which can be tailored to specific needs from two channels with one M8041A to up to 10 channels.

The modules must be installed in the M9514A AXIe 14-slot chassis as shown in Table 3 on page 17:

<table>
<thead>
<tr>
<th>Slot Number</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td># 1</td>
<td>For M8030A-BU1, M9537A AXIe embedded controller. For M8030A-BU2, this slot is empty and covered with filler front-panel</td>
</tr>
<tr>
<td># 2-4</td>
<td>M8041A module</td>
</tr>
<tr>
<td># 5-6</td>
<td>M8051A module</td>
</tr>
<tr>
<td># 7</td>
<td>M9521A AXIe system module, always included in M8030A-BU1 or M8030A-BU2, mandatory</td>
</tr>
<tr>
<td># 8-9</td>
<td>M8051A module</td>
</tr>
</tbody>
</table>
Figure 5 on page 18 shows an example of modules arrangement in the M9514A AXIe 14-slot chassis.

<table>
<thead>
<tr>
<th>Slot Number</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td># 10-11</td>
<td>M8051A module</td>
</tr>
<tr>
<td># 12-13</td>
<td>M8051A module</td>
</tr>
<tr>
<td># 14</td>
<td>M8192A multi-channel synchronization module, mandatory</td>
</tr>
</tbody>
</table>

Figure 5 Example of M8030A module arrangement
M8040A Modular Configuration

The M8040A is a modular test solution which simplifies accurate receiver characterization of devices operating up to 32 and 64 GBaud with NRZ and PAM4. It supports the following modules.

- M8045A pattern generator
- M8046A analyzer
- M8057A remote head

The following section describes and illustrates various setup combinations in which you can install the M8040A modules.

M8045A Pattern Generator Module

The M8045A module can be a one or two data channel system (a second channel can be added with license). A one channel instrument has to be returned to the factory for installing the second channel (hardware) and license. It occupies three slots of the 5-slot M9505A AXIe chassis. The following figure illustrates an M8045A module two data channel system) installed in an M9595A AXIe chassis.

![M8045A Module (2 Channels)](image)

**NOTE**

The M8045A module must be installed in slots 1 through 3 of the M9505A AXIe chassis for proper local bus communication unless the M9537A AXIe Embedded Controller is installed (must be in slot 1).
M8046A Analyzer Module

The M8046A module occupies single slot of the 5-slot M9505A AXIe chassis. The following figure illustrates an M8045A module with M8046A module installed in an M9505A AXIe chassis.

![M8046A Module](image1)

Figure 7 M8045A and M8046A configuration

M8057A Remote Head

The M8057A remote head is an external box which can be connected to each channel of M8045A module, using the matched pair of cables. It helps in minimizing signal degradations caused by lossy channels. The following figure illustrates an M8057A remote head connected with one channel of M8045A module.

![M8057A Remote Head](image2)

Figure 8 M8045A, M8046A and M8057A configuration
Using Identifiers

There are three different types of Identifiers:

- Location
- Group Name
- Measurement Name

Location Identifiers

When an instrument has several output channels with identical capabilities or a subset of capabilities and has more than one module, the selection of which module/channel to use is done using its location identifier.

The location identifier is a predefined notation that can be used to address certain properties, like high level, low level etc., of channels. The location identifier is surrounded by single quotes (‘identifier’) or double quotes (“identifier”). A location identifier will be appended to a SCPI command followed by its argument(s). The location identifier addresses the corresponding components. The location identifier consists of two parts: the first part specifies the module, the second part addresses, in most cases, a visible ‘component’ of a module, like an input or output connector.

For example, SCPI command :OUTP ‘M1.DataOut1’, OFF turns channel 1 (DataOut1) of module 1 (M1) output off.

If a SCPI command is to be sent to a specific module only and not a specific channel, simply use “M1” or “M2” as the location identifier.

If you want to set a value that acts on all components of a modular instrument (for example, set all jitter sources to off), omit the location identifier.

Multiple Source Per Channel Identifier

Some features have multiple sources per channel. To identify these sources, the “(*)” suffix is used.

For example, there are two periodic jitter sources per channel in a module. The following SCPI command specifies periodic jitter source 2 as the source for module 1/data out channel 1:

[:SOURce]:JITTer:PERiodic2[:STATe] ’m1.dataout1’,ON
Group Name Identifiers

Group name identifiers are used to affect a group of inputs or outputs using a single SCPI command. That is, all properties belonging to these identifiers can be addressed simultaneously using the group name.

The following SCPI command shows the syntax for creating a group name then adding the desired location identifiers.

```
SYSTem:INSTrument:GROup:DEFine
    'GroupName', 'Identifier', 'Identifier', 'Identifier', '...' 
```

The following is an example showing 'Outputs' as the group name followed by 'M1.DataOut1' and 'M1.DataOut2' location identifiers assigned to the 'Outputs' group name identifier.

```
```

The following SCPI command will set all voltage amplitudes in the 'Outputs' group to 0.05.

```
[:SOURce]:VOLTage[:AMPLitude]VOLT:AMPL 'Outputs', 0.05
```

Measurement Name Identifiers

Measurement name identifiers are used to set up parameters associated with a specific measurement name. For example, the following shows the bit error ratio plugin SCPI command used to define a measurement name called 'MyMeasurement':

```
:PLUGin:ERATio:NEW 'MyMeasurement'
```

You can now set up parameters within the :PLUGin:ERATio subsystem to be associated with the 'MyMeasurement' identifier.

The following shows the accumulation duration mode (set to fixed time) and duration (set to 120 seconds) parameters associated with the 'MyMeasurement' identifier.

```
```
The M8020A/M8030A/M8040A platform supports a plugin interface. C# assemblies which implement certain interfaces are recognized by the software and integrated into the M8070B GUI and instrument software. Error ratio, Output Timing and Jitter Tolerance measurements are examples of the plugin concept. These C# assemblies are placed in a predefined location and are integrated into the M8070B software at start up time automatically.

These measurements can be controlled by the GUI and by SCPI commands. This means that the SCPI tree has to be defined by the plugin itself and is integrated into the existing SCPI tree of the M8020A/M8030A/M8040A platform. SCPI commands for a plugin are placed below a predefined node named ":PLUGin" followed by the type of the plug in.

There is a set of "general" SCPI commands for creating, deleting and controlling a plugin. Also, there is a flexible set of SCPI commands provided by the plugin itself for setting and reading measurement parameters.

Additionally, a user can create multiple measurements of the same type. A measurement name identifier is used to address a created measurement.
2 Recommended Programming Techniques

- Output Protection / 26
- Controlling the Output Levels / 27
- Reading the M8020A/M8030A/M8040A's Status / 28
- Reading the M8020A/M8030A/M8040A's Status / 28
- Working with Patterns / 36
- Creating Pattern Sequences / 43
Output Protection

Pattern Generator Output Termination

The pattern generator’s output ports must be terminated with 50 Ω if they are not connected. Termination of output ports improves the test performance.

> **NOTE**
> Refer also to the M8020A/M8030A/M8040A Getting Started Guide for information on terminating the M8061A outputs to ensure proper performance.

Output Protection Algorithm

The instrument has an internal protection algorithm that protects the instrument from improper termination of the pattern generator’s output ports.

The algorithm checks whether the termination voltage and termination resistance are correct before the output is enabled. This check happens once only.

In non-balanced dc coupled mode, the termination voltage must be within ± 0.1 V of the specified value. The resistance must be between 40 and 60 Ω.

In balanced mode, the resistance must be between 80 and 120 Ω (differential termination).

In ac coupled mode, which is intended for balanced only (for example, 8b/10b coded patterns), the output is enabled if no termination voltage and no termination resistance are detected. This is the case if ac capacitors are in front of the outputs. The check distinguishes between a terminated or an open output.

> **NOTE**
> Do not operate the output driver into an open.
Controlling the Output Levels

Controlling the Output Levels – Concepts

When the output levels are changed at the M8020A/M8030A/M8040A data and clock output ports, four parameters are changed:

- $V_{\text{high}}$
- $V_{\text{low}}$
- $V_{\text{ampl}}$
- $V_{\text{offs}}$

The M8020A/M8030A groups these parameters into "pairs" ($V_{\text{ampl}}/V_{\text{offs}}, V_{\text{high}}/V_{\text{low}}$). If one of these values is modified, its "partner" remains constant and the values in the other pair are modified accordingly. For example, if $V_{\text{ampl}}$ is changed, $V_{\text{offs}}$ stays constant and $V_{\text{high}}$ and $V_{\text{low}}$ are modified accordingly.

Controlling the Output Levels – Procedures

Changing the Voltages with SCPI

The following commands show how you would set the data output so that it has an amplitude of 1 V and an offset of 0.5 V:

```
:SOUR:VOLT:AMPL 'M1.DataOut1',1; OFFS 'M1.DataOut1',0.5
```

This sets the output accordingly ($V_{\text{high}} = 1$ V, $V_{\text{low}} = 0$ V).

Conversely, you could set $V_{\text{high}}$ and $V_{\text{low}}$ directly:

```
:SOUR:VOLT:HIGH 'M1.DataOut1',1;
LOW 'M1.DataOut1',0
```
Reading the M8020A/M8030A/M8040A’s Status

Reading the M8020A/M8030A/M8040A’s Status – Concepts

The M8020A/M8030A/M8040A has a set of status registers that you can use to monitor the status of the hardware, software and any running tests.

Overview of Registers

Specifically, it has the following registers:

- Status Byte
  The Status Byte is a single register that stores the events occurring in the other registers.

- Standard Event Status Register
  The Standard Event Status Register monitors some non-critical errors and basic operations.

- Questionable Data Status Register
  The bits in the Questionable Data Status Register are set when certain events occur in the M8020A/M8030A/M8040A that can lead to questionable results.

- Operation Status Register
  The Operation Status Register indicates when certain operations have been completed.

How the M8020A/M8030A/M8040A Uses Status Registers

You can determine the state of certain instrument hardware and firmware events and conditions by programming the status register system.

The following subsections provide you with details about the M8020A/M8030A/M8040A’s status system.

Overview of the M8020A/M8030A/M8040A’s Status System

The M8020A/M8030A/M8040A has status reporting features that give important information about events and conditions within the instrument. For example, a flag may be set to indicate the end of a measurement or perhaps a command error. To access this information, it is necessary to query a set of registers using SCPI.
M8020A/M8030A/M8040A Status System Structure

The M8020A/M8030A/M8040A's status system is comprised of multiple registers that are arranged in a hierarchical order. The lower-level status registers propagate their data to the higher-level registers in the data structures by means of summary bits. The Status Byte register is at the top of the hierarchy and contains general status information for the M8020A/M8030A/M8040A's events and conditions. All other individual registers are used to determine the specific events or conditions.

Status Register Group Model

Figure 9 on page 29 illustrates the typical structure of a status register.

The M8020A/M8030A/M8040A instrument status register model follows the structure described in IEEE 488.2, section 11.4.2.

A condition is a device state which is either TRUE or FALSE. A condition register reflects these states in its condition bits.

A Condition Register is embedded in a register structure consisting of Transition Filters, an Event Register and an Event Enable Register.

![Status register structure diagram](image)
This register is read by the CONDition? SCPI commands.

**Transition Filters:** A positive transition filter allows an event to be reported when a condition changes from false to true. A negative transition filter allows an event to be reported when a condition changes from true to false. Setting both positive and negative filters true allows an event to be reported anytime the condition changes. Clearing both filters disables event reporting.

This register is set and read by the NTRansition[?] and PTRansition[?] SCPI commands.

**Event Register:** An event register latches transition events from the condition register as specified by the positive and negative transition filters. Bits in the event register are latched and once set, they remain set until cleared by either querying the register contents or sending the *CLS command.

This register is read by the EVENt? SCPI commands.

**Event Enable Register:** An enable register specifies the bits in the event register that can generate a summary bit. Summary bits are, in turn, used by the next higher register.

This register is set and read by the ENABle[?] SCPI commands.

The registers work together as follows:

1. The **Condition Register** corresponds to a condition on the hardware or in the software. If the monitored condition is present, the corresponding bit is high.

2. The **Transition Registers** monitor changes in the Condition Register. If the Positive Transition Register is configured to watch for a condition, when this condition occurs and the bit in the Condition Register goes high, the Positive Transition Register passes this event to the **Event Register**.

3. If this bit is enabled in the **Event Enable Register**, a summary bit is generated in the next higher register. For the higher register, this is the Condition Register and the event is handled the same way as described here.
M8020A/M8030A/M8040A Status Reporting Structure

Figure 10 on page -31 shows the M8020A/M8030A/M8040A’s status reporting structure.
Status Byte

The Status Byte is the summary register to which the other registers report. Each reporting register is assigned a bit in the Status Byte Register. The bits in the Status Byte byte have the following meaning:

Table 4  Status byte descriptions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>EAV</td>
<td>Error available: the error queue contains at least one message.</td>
</tr>
<tr>
<td>3</td>
<td>QUES</td>
<td>A bit has been set in the Questionable Data Status register (indicates that a signal is of questionable quality).</td>
</tr>
<tr>
<td>4</td>
<td>MAV</td>
<td>Message available: There is at least one message in the message queue.</td>
</tr>
<tr>
<td>5</td>
<td>ESB</td>
<td>A bit in the Standard Event Register has been set.</td>
</tr>
<tr>
<td>6</td>
<td>RQS</td>
<td>ReQuest Service.</td>
</tr>
<tr>
<td>7</td>
<td>OPER</td>
<td>A bit in the Operation Status Register has been set.</td>
</tr>
</tbody>
</table>
Standard Event Status Register

The Standard Event Status register provides general-purpose information about the instrument. It sets bit 5 in the Status Byte.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation Complete</td>
<td>Operation Complete bit. It is set in response to the *OPC command, but only if the instrument has completed all its pending operations.</td>
</tr>
<tr>
<td>1</td>
<td>Request Control</td>
<td>Request Control bit.</td>
</tr>
<tr>
<td>2</td>
<td>Query Error</td>
<td>Query error bit. It indicates that there is a problem with the output data queue. There has been an attempt to read the queue when it is empty, the output data has been lost, or the query command has been interrupted.</td>
</tr>
<tr>
<td>3</td>
<td>Device Dependent Error</td>
<td>Device-dependent error bit. It is set when an instrument-specific error has occurred.</td>
</tr>
<tr>
<td>4</td>
<td>Execution Error</td>
<td>Execution error bit. It is set when a command (GPIB instrument specific) cannot be executed due to an out of range parameter or some instrument condition that prevents execution.</td>
</tr>
<tr>
<td>5</td>
<td>Command Error</td>
<td>Command error bit. It is set whenever the instrument detects an error in the format or content of the program message (usually a bad header, missing argument, or wrong data type etc.).</td>
</tr>
<tr>
<td>6</td>
<td>User Request</td>
<td>User Request bit.</td>
</tr>
<tr>
<td>7</td>
<td>Power On</td>
<td>Power-on bit. It is set each time the instrument is powered from off to on.</td>
</tr>
</tbody>
</table>
**Questionable Status Register**

The Questionable Status Register indicates that a currently running process is of questionable quality. The output of this register sets bit 3 of the Status Byte.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 8</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>INSTRument</td>
<td>This bit indicates the status of certain hardware components.</td>
</tr>
<tr>
<td>10</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>PROTection</td>
<td>This bit indicates whether one or more of the instrument outputs detect an overload condition.</td>
</tr>
<tr>
<td>12</td>
<td>SYMBol</td>
<td>This bit indicates whether the error detector(s) has experienced a symbol alignment loss.</td>
</tr>
<tr>
<td>13 - 15</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**Operation Status Register**

The output of this register gives information about the current operation the instrument is performing. It sets bit 7 of the Status Byte.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 7</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>RUN</td>
<td></td>
</tr>
<tr>
<td>9 - 15</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
Program Snippet Extending the *opc? Functionality

Whenever changes are done that require the hardware to stop clock generation, a "*opc?" command does not wait long enough. It only waits until the clock generation is stopped and not when the clock generation is again started. You have to wait until the hardware is in run condition again (clocks are again running). In general, once you perform these kind of operation, you should execute "*opc?" command to make sure whether the operation is completed or not.

This can be polled in the status system, e.g. with a program snippet like this:

```csharp
var timeStart = DateTime.Now;
while ((response != "1") && ((DateTime.Now - timeStart).TotalSeconds < 60))
Assert.AreEqual("1", response);
```

This program snippet can be used after the following operations:

- Frequency change or anything related. Everything that is in the M8070B GUI under "ClkGen" and "Synthesizer" functional blocks.
- SSC parameter change (everything that is in the M8070B GUI under "ClkGen", "SSC" functional blocks or the global SSC on/off)
- Changes in the sequence symbol width
- *rst or any other reset command or setting load, as they change all parameters.
Working with Patterns

Patterns can be accessed with a program and are stored in three different areas:

1. Local to current setting ("current:")
2. Shared between settings ("shared:")
3. Factory supplied standard patterns ("factory:"). These patterns are read only and cannot be modified.

Below these root nodes is a folder structure using '/' as a separation character. So a complete pattern name might be something like "factory:SATA/LTDP-short".

Patterns consist of a sequence of symbols. A symbol has a coding:

1. Binary ("Bit")
2. 8b10b ("B8B10")
3. 128/130 ("B128B130")
4. 128/132 ("B128B132")

A symbol can have additional attributes to modify behavior. All symbols support:

1. Mask: this attribute affects DataIn locations only. It specifies if the symbol is actually compared or masked (excluded from compare).
2. Squelch: this attribute affects DataOut locations only. If this attribute is 1, a squelch (out of band) level is used.

In addition, there are coding specific attributes, to control aspects of the coding such as bypassing or using a scrambler. Refer to :DATA:PATTern:IDATa[?] on page 394 for more information.

The :DATA:PATTern:USE SCPI command is used to specify the patterns and attributes. Refer to :DATA:PATTern:USE[?] on page 393 for details on how to specify a pattern and set up its parameters. The parameters include the pattern name, number of symbols, symbol coding (optional), mask (optional) and squelch (optional).
Symbol Coding Descriptions

Bit Coding

To encode a bit coded (binary) symbol 1, 2, or 3 bits are needed. This depends on the use of mask and squelch.

<table>
<thead>
<tr>
<th>Table 8</th>
<th>Plain bit coding without using mask or squelch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Offset/Range</td>
<td>Bit (range) Name</td>
</tr>
<tr>
<td>0</td>
<td>Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 9</th>
<th>Mask is used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Offset/Range</td>
<td>Bit (range) Name</td>
</tr>
<tr>
<td>0</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Mask</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 10</th>
<th>Squelch is used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Offset/Range</td>
<td>Bit (range) Name</td>
</tr>
<tr>
<td>0</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Squelch</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 11</th>
<th>Mask and squelch are used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Offset/Range</td>
<td>Bit (range) Name</td>
</tr>
<tr>
<td>0</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Mask</td>
</tr>
<tr>
<td>2</td>
<td>Squelch</td>
</tr>
</tbody>
</table>
Table 12  Error is used

<table>
<thead>
<tr>
<th>Bit Offset/Range</th>
<th>Bit (range) Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Data</td>
<td>Data bit</td>
</tr>
<tr>
<td>1</td>
<td>Error</td>
<td>Error (ignored on DataOut, ignored on DataIn)</td>
</tr>
</tbody>
</table>
### 8b10b Coding

To encode an 8b10b symbol 16 bits (2 bytes) are used.

#### Table 13  8b10b coding

<table>
<thead>
<tr>
<th>Bit Offset/Range</th>
<th>Bit (range) Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>Data</td>
<td>Symbol data</td>
</tr>
<tr>
<td>8</td>
<td>K/D</td>
<td>$0 = D$-character, $1 = K$-character</td>
</tr>
<tr>
<td>9</td>
<td>Reserved for future use. Must be set to 0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Mask</td>
<td>Mask (if present, ignored on DataOut)</td>
</tr>
<tr>
<td>11</td>
<td>Squelch</td>
<td>Squelch (if present, ignored on DataIn)</td>
</tr>
<tr>
<td>12</td>
<td>Enable scrambler</td>
<td>Enable scrambler (ignored on DataIn)</td>
</tr>
<tr>
<td>13</td>
<td>Pause scrambler</td>
<td>Pause scrambler (ignored on DataIn)</td>
</tr>
<tr>
<td>14</td>
<td>Reset scrambler</td>
<td>Reset scrambler (ignored on DataIn)</td>
</tr>
<tr>
<td>15</td>
<td>Start of Frame</td>
<td>Start of Frame (ignored on DataOut)</td>
</tr>
</tbody>
</table>

#### Table 14  Variant for captured pattern

<table>
<thead>
<tr>
<th>Bit Offset/Range</th>
<th>Bit (range) Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>Data</td>
<td>Symbol data</td>
</tr>
<tr>
<td>8</td>
<td>K/D</td>
<td>$0 = D$-character, $1 = K$-character</td>
</tr>
<tr>
<td>9</td>
<td>Reserved for future use. Must be set to 0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Error</td>
<td>Error (ignored on DataOut, ignored on DataIn)</td>
</tr>
<tr>
<td>15:11</td>
<td>Squelch</td>
<td>Reserved for future use. Must be set to 0</td>
</tr>
</tbody>
</table>
128/130 Coding

To encode 128/130 symbol 144 bits (18 bytes) are used.

<table>
<thead>
<tr>
<th>Bit Offset/Range</th>
<th>Bit (range) Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:0</td>
<td>Framing</td>
<td>Framing bits</td>
</tr>
<tr>
<td>129:2</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>130</td>
<td>Mask</td>
<td>Mask (if present, ignored on DataOut)</td>
</tr>
<tr>
<td>131</td>
<td>Squelch</td>
<td>Squelch (if present, ignored on DataIn)</td>
</tr>
<tr>
<td>132</td>
<td>Reset Scrambler</td>
<td>Reset scrambler (ignored on DataIn)</td>
</tr>
<tr>
<td>133</td>
<td>Pause scrambler</td>
<td>Pause scrambler (ignored on DataIn)</td>
</tr>
<tr>
<td>134</td>
<td>Bypass scrambler</td>
<td>Scrambler bypass</td>
</tr>
<tr>
<td>135</td>
<td>Bypass byte 0 scrambler</td>
<td>Scrambler bypass byte 0</td>
</tr>
<tr>
<td>136</td>
<td>Do DC balancing</td>
<td>Do DC balancing (ignored on DataIn)</td>
</tr>
<tr>
<td>137</td>
<td>Reset DC balancing</td>
<td>Reset DC balancing (ignored on DataIn)</td>
</tr>
<tr>
<td>138</td>
<td>Send scrambler state</td>
<td>Send scrambler state</td>
</tr>
<tr>
<td>139</td>
<td>Reset Parity</td>
<td>Reset Parity (ignored on DataIn)</td>
</tr>
<tr>
<td>140</td>
<td>Pause Parity</td>
<td>Pause Parity (ignored on DataIn)</td>
</tr>
<tr>
<td>141</td>
<td>Start of Frame</td>
<td>Start of Frame (ignored on DataOut)</td>
</tr>
<tr>
<td>143:142</td>
<td></td>
<td>Reserved for future use. Must be set to 0</td>
</tr>
</tbody>
</table>
128/132 Coding

To encode a 128/132 symbol 144 bits (18 bytes) are used.

### Table 16  128/132 coding

<table>
<thead>
<tr>
<th>Bit Offset/Range</th>
<th>Bit (range) Name</th>
<th>Coding-128/132 content</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:0</td>
<td>Framing</td>
<td>Framing bits</td>
</tr>
<tr>
<td>131:4</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>132</td>
<td>Mask</td>
<td>Mask (if present, ignored on DataOut)</td>
</tr>
<tr>
<td>133</td>
<td>Squelch</td>
<td>Squelch (if present, ignored on DataIn)</td>
</tr>
<tr>
<td>134</td>
<td>Reset Scrambler</td>
<td>Scrambler reset (ignored on DataIn)</td>
</tr>
<tr>
<td>135</td>
<td>Pause Scrambler</td>
<td>Scrambler pause (ignored on DataIn)</td>
</tr>
<tr>
<td>136</td>
<td>Bypass Scrambler</td>
<td>Scrambler bypass</td>
</tr>
<tr>
<td>137</td>
<td>Bypass Byte 0 Scrambler</td>
<td>Scrambler bypass byte 0</td>
</tr>
<tr>
<td>138</td>
<td>Do DC Balancing</td>
<td>Do DC balancing (ignored on DataIn)</td>
</tr>
<tr>
<td>139</td>
<td>Reset DC Balancing</td>
<td>Rest DC balancing (ignored on DataIn)</td>
</tr>
<tr>
<td>140</td>
<td>Send Scrambler State</td>
<td>Send scrambler state (ignored on DataIn)</td>
</tr>
<tr>
<td>141</td>
<td>Reset Parity</td>
<td>Reset parity (ignored on DataIn)</td>
</tr>
<tr>
<td>142</td>
<td>Pause Parity</td>
<td>Pause parity (ignored on DataIn)</td>
</tr>
<tr>
<td>143</td>
<td>Start of Frame</td>
<td>Start of frame (ignored on DataOut)</td>
</tr>
</tbody>
</table>
Symbol Sequences

When combining symbols to a sequence of symbols they are filled in one after the other without any gaps. For the non-bit codings that use multiple bytes, the symbols naturally start again on a byte boundary. For the bit codings (binary) they start at the next available bit. Bits are used in a fashion that allows reading from “left to right”. This means the symbol bit 0 is encoded as bit 7 in a byte.

Figure 11  Example 1: plain bit coding

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure 12  Example 2: mask and squelch bit coding

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 0 1 0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0 1 0 1 0 1 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 1 0 1 0 1 0 1</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0 1 0 1 0 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Creating Pattern Sequences

A pattern sequence is configured using a language. The pattern sequence language is based on XML.

Sequence Editor

Pattern sequences are created and edited using the Sequence Editor in the M8070B software interface as shown in Figure 14 on page -43.
A default “Generator” and “Analyzer” sequence are configured in the Sequence Editor as shown in Figure 14 on page -43. You can edit these sequences or create new ones.

As a pattern sequence is being configured, the corresponding XML elements are generated and can be viewed in the XML editor by clicking on the <Xml> button shown in Figure 15 on page -44.

Figure 15 <Xml> button

Figure 16 on page -44 shows the XML elements for the default Generator and Analyzer. Clicking the UI button will return to the graphical interface.

Figure 16 XML view

In UI view, click in a pattern sequence block to display its sequence settings as shown in Figure 17 on page -45.
Pattern Sequence Building Blocks

**Minimal Sequence**

A minimalistic sequence looks like the following:

```xml
<?xml version="1.0" encoding="utf-16"?>
<sequenceDefinition
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.keysight.com/schemas/M8000/DataSequence">
</sequenceDefinition>
```

This sequence generates a 0 signal to all assigned locations.

**Description**

To specify a description for the sequence, insert the following:

```xml
<description>Default 2^7-1 PRBS Sequence</description>
```

**Version Number**

To specify a version number for the sequence syntax, insert the following:

```xml
<version>1.0.0</version>
```
Symbol Width

To specify the symbol encoding this sequence works with, insert the following:

```xml
<symbolWidth>1</symbolWidth>
```

Available symbol widths are:
- 1: this is for bit coded symbols
- 10: this is for 8b/10b coded and bit coded symbols
- 130: for 128/130 coded and bit coded symbols
- 132: for 128/132 coded and bit coded symbols

Pattern Generation Granularity

Pattern generation has granularity. A single bit cannot be generated. Instead, multiple bits are generated at once. Compared to other BERTs like the J-BERT N4903B, different granularities are supported to allow for less restrictions in the pattern generation. Also, granularity is hidden as much as possible by unrolling patterns internally.

<table>
<thead>
<tr>
<th>Symbol Width</th>
<th>Granularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>130</td>
<td>130</td>
</tr>
<tr>
<td>132</td>
<td>132</td>
</tr>
</tbody>
</table>

The actual pattern sequence attributes are specified within:

```xml
<sequence replicate="Serialize"></sequence>
```

Replicate allows global selection of how to distribute PRBS or memory patterns to multiple locations.
- Serialize: first bit/symbol is assigned to first location, next bit/symbol to next location and so on. This means a pattern needs to have a minimum block length * locations bits. In the case of a PRBS, a PRWS is issued, setting the phases of the PRBS generators in a way that they will form a PRBS after a MUX.
- Copy: each location gets a copy of the PRBS /pattern.
Blocks and Loops

The basic building blocks of a sequence are blocks and loops.

A block is the basic unit of a sequence. Here patterns are assigned to locations or branches and are specified to other blocks. A block transitions to the next as a default action. But it also might be enclosed in a loop. Nested loops are also possible. There are three counted loop levels plus an infinite loop level:

- Loop level 1 can be used to repeat a block.
- Loop level 2 and 3 can span multiple blocks, but it must be ensured that either the start or end of a loop is on a fresh block.
- Loop level 4 is for infinite loops.

The following illustrates valid loop nesting with the full 4 loop levels:

Figure 18 Loop nesting

Count attribute specifies the number of loop iterations. The valid value range is 1 to $2^{32}$. If omitted, infinite is assumed. The enabled attribute allows the complete loop, including all loop content, to be temporarily disabled. If omitted, the default is enabled.

```
<loop count="1" enabled="1"></loop>
```
Length Attribute

The length attribute specifies the “length” of the block in number of bits. The enabled attribute allows the block to be temporarily disabled (ignored). If omitted, the default is enabled.

```xml
<block length="1" enabled="1"></block>
```

Synchronization and Loop Block

A special combination of sync and loop is available for achieving synchronization:

```xml
<syncAndLoopBlock length="100" enabled="1"></syncAndLoopBlock>
```

Other attributes for block or syncAndLoopBlock:

1. name: for naming a block.
2. cdr: for freezing CDR when CDR control is “sequence”.
3. errorInsertion: for suppressing error insertion.
4. disparity: for setting the disparity in 8b10b coding.
5. startWordAlign: for initiating a word alignment in 8b10b and 128/130 coding. In the syncAndLoopBlock this flag is used for the initial block only, not for the looped block.

Pattern Selection

Within a block or syncAndLoopBlock, select the type of patterns to use. The choices are clock, pulse, Prbs, static and memory pattern. The XML elements for pattern selection are as follows:

```xml
<prbs polynomial="2^7-1" seed="127" invert="0" replicate="Serialize" enabled="1"></prbs>
```

```xml
<pattern source="factory:SATA/LTDP-short_b8b10" replicate="Serialize" enabled="1"></pattern>
```

```xml
<static value="0" compare="1" />
```

```xml
<pulse offset="0" width="10" compare="1" squelch="1"/>
```

```xml
<clock dividers="2" compare="1"/>
```

```xml
<linkTraining direction="up"/>
```
Recommended Programming Techniques

Sequencer Triggers

Trigger output pulses can be generated and controlled from sequence blocks. These pulses are sent to the front panel TRIG OUT connectors.

This requires that the trigger mode be set to sequencer controlled. The following describes how to change the trigger mode.

1 In the M8070B software interface, select System > Module View.
2 Click on Trig Out.
3 Under Configuration parameters, change Operating Mode to Sequencer Controlled.

Refer to :OUTPut:TRIGger:MODE[?] on page 237 for information about setting the trigger mode using this SCPI command.

Two variants are available:

1 A trigger pulse can be generated relative to a sequence block.
2 A trigger pulse can be generated whenever a specific bit sequence is generated by a PRBS generator.

This is controlled via nodes within a block node.

NOTE

If multiple sequences are used, only one sequence can actually have control of the trigger.

Triggering Relative to a Sequence Block

The following is an example of a trigger pulse relative to a sequencer block:

<trigger>

  <pulse offset="0" width="32"/>

</trigger>

The offset can be omitted and defaults to 0. The width can also be omitted and defaults to 32.
Recommended Programming Techniques

Triggering Based on Specific Bit Sequence

The following is an example of a trigger pulse generated based on a specific PRBS bit sequence:

```
<trigger>
  <pulseOnPrbsMatch width="32" matchPattern="0000000" polynomial="2^7-1" seed="127" invert="0"/>
</trigger>
```

All attributes can be omitted. The width defaults to 32. The match pattern defaults to all 0’s or all 1’s depending on the invert attribute. The polynomial defaults to the polynomial used by the generators or analyzers. Invert also defaults to the PRBS invert setting of the generators or analyzers.

Block Branching

Within a block a conditional branch to another block can be done. The most simple form is as follows:

```
<if source="Break"/></if>
```

This example simply jumps to the next block when a break command is done. The following describes how to set up a break command.

1. In the M8070B software interface, select Patterns > Sequence Editor.
2. Under the Block Branches settings, change Condition to Break.

Refer to :DATA:SEQUence:BREak on page 406 for information on setting the break command using this SCPI command.

When a branch to a different block is desired, use the “goto” attribute:

1. In the M8070B software interface, select Patterns > Sequence Editor.
2. Under the Block Branches settings, click in the Go To Block field and enter the block name.

Other potential conditions are SysInA, SysInB, CtrlInA, or CtrlInB. Break, SysInA and SysInB are global events and affect all locations of a sequence. CtrlInA and CtrlInB are local to a module and are most suitable when all locations of a sequence are using a single module.
It is also possible to specify a reaction based on a static level or a transition. Using the “event” attribute, you can select “low”, “high”, “positiveEdge” and “negativeEdge”.

1. In the M8070B software interface, select **Patterns > Sequence Editor**.
2. Under the **Block Branches** settings, change **Transition** to **Positive**, **Negative**, **Static High**, or **Static Low**.

**Link Training Events**

- **Link Training Target State**: indicates that the Link Training is in the target state (that is loopback)
  
  ```xml
  <if source="LinkTrainingTargetState"/>
  ```

- **Link Training Detect State**: indicates that the Link Training is in the detect state (that is the reset state)
  
  ```xml
  <if source="LinkTrainingDetectState"/>
  ```

- **Link Training Error**: indicates that an error occurred when trying to bring up the link (that is Link Training reached the target state but not the target speed)
  
  ```xml
  <if source="LinkTrainingError"/>
  ```

**Block Controls**

Within a block it can be specified to generate a signal on CTRL OUT A, SYS OUT A, or SYS OUT B front panel connector. This can be useful for triggering an external device such as an oscilloscope or to generate auxiliary signals such as a laser enable signal. However, be aware that there is no specified timing relationship to the DataOut.

```xml
<control sink="CtrlOutA" sinkLocation="M1" value="pulse"/>
```

In a two module configuration, the sinkLocation attribute is used to specify on what module the CTRL OUT A signal is generated.

The shape of the signal is specified with the value attribute. It can be a static 0, static 1, or a pulse.

**NOTE**

SysOutA and SysOutB are internally controlled from the Trigger Sequencer.

Therefore, these signals can only be used in the same sequence where the trigger out is controlled.
Global Block Controls

CtrlOutA can only be used if it is not currently being used by the global block controls.

Global block controls are used to set up global routing of signals. Signals can be routed to the CTRL OUT A front panel connector only.

One common use could be for generating a pulse on CTRL OUT A whenever an error on the M1.DataIn1 or M1.DataIn2 of the corresponding module occurs:

```xml
<controls>
  <control sink="CtrlOutA" source="Error"
    sourceLocation="M1.DataIn1"/>
</controls>
```

With the additional “signalShape” attribute, it can be selected if a 0/1 pulse is generated whenever an error occurs.

Coding Configuration

To select the physical layer protocol, the following node can be added:

```xml
<phyProtocol name="PCIe3" />
```

The available PHY protocol choices are as follows:

- None (default)
- CCIX
- PCIe2
- PCIe3
- PCIe4
- PCIe5
- USB 3.0
- USB 3.1
- SATA3
- SAS2
- SAS3
These choices are further restricted by the module being used, and the installed licenses on the modules.

For M8020A, only the PHY protocols for which an LTSSM is available are supported (PCIe3, PCIe4, USB 3.0, and USB 3.1).

For M8045A, the allowed PHY protocols are PCIe3, PCIe4, and PCIe5.

For M8046A the availability is controlled by the licenses M8046A-0S2, M8046A-0S4 and M8046A-0S6 and the corresponding upgrade and trial licenses.

The PHY protocol “None” is always available.

On M8040A modules, the PHY protocol selection does choose the coding specific configurations automatically and it will override any user configurations.

The M8040A modules do only support a symbolWidth of 1, and bit coded patterns. They do not support the coded patterns like M8020A.

Therefore, the patterns must contain a continuous sequence of bits as required by the PHY protocol. All protocol specific encoding and validity rules must be provided by the pattern.

Additionally the following requirements exist on M8040 setups:

- The patterns for M8045A must contain the align, filler, SKP OS symbol sequences as required by the standard being tested.
  - For 8b10b patterns the pattern must contain the 10b symbols with a correct running disparity.
- The patterns for M8046A may not contain align, filler, SKP OS symbol sequences.
  - For 8b10b patterns, the pattern must contain the 10b symbols, each symbol normalized to negative disparity.
- The granularity requirements are defined by the channel being used (512 bit for M8040A) and the PHY protocol that is selected. This leads to the requirement that in the pattern sequence (block length * loop count) must be a multiple of LeastCommonMultiple (512, protocol granularity).

The PHY protocol granularity is as follows:

- 10 for 8b10b
- 130 for 128b130b
- 132 for 128b132b
- 150 for 128b150b
For sequence blocks that do not match the granularity requirement, the loop count can be calculated as such:

- Pattern Granularity = LeastCommonMultiple (512, PHY protocol granularity).
- Required Block Length = LeastCommonMultiple (Pattern Granularity, Pattern Length).
- Loop count = Required Block Length / Pattern Length.
- The M8046A does support algorithmic PRBS patterns only when the PHY protocol is set to None.

On M8020A setups, various aspects of coding can be configured. This is done with the `<codingConfigurations>` node. It is inserted just before the `<symbolWidth>` node. Within this node the 8b10b specific attributes are configured in a `<b8b10>` node. The align symbol is configured with an optional attribute. If not specified the default align symbol is K28.5. Also, you can specify a substitution symbol in an attribute that is inserted into the received data stream when an invalid code word is recognized. The default for this is K28.0. So the complete node would look like the following:

```xml
<b8b10 align="K28.5" substitution="K28.0"/>
```

And can be shortened to the following:

```xml
<b8b10/>
```

Up to four filler primitives can be specified. Each filler primitive can consist of up to 4 symbols, where a symbol can be any valid 8b10b symbol or a '*' that is used as a wildcard matching any symbol.

A filler symbol specification looks like the following:

```xml
<fillerPrimitive symbol1="K28.1" symbol2="K28.1"/>
```

After specifying a filler primitive, various aspects of the (de)scrambler can be configured with the `<scrambler>` node. With the polynomial attribute the USB/PCIe or SATA polynomial can be selected. The reset value for the scrambler and the reset value for the scrambler after a hold sequence can be configured with the `resetValue` and `resetValueAfterHold` attribute. In addition, it can be specified if the scrambler is reset after a filler primitive was removed with the `resetAfterFillerRemove` attribute. All these nodes are optional, with defaults suitable for USB testing. If just `<scrambler>` node is given without attributes, it is equivalent to the following:

```xml
<scrambler polynomial="USB/PCIe" resetValue="65515" resetValueAfterHold="6143" resetAfterFillerRemove="false"/>
```
Within the scrambler node, sequences of up to 4 symbols can be configured that control the scrambler when they are seen in the incoming data stream. With resetPrimitive the scrambler is reset, with holdPrimitive it is paused and with pauseStartPrimitive it is paused until a pauseEndPrimitive is seen.

The following are examples for the different standards.

**USB 3.0**

```xml
<codingConfigurations>
  <b8b10>
    <fillerPrimitive symbol1="K28.1" symbol2="K28.1"/>
    <scrambler>
      <resetPrimitive symbol1="K28.5"/>
    </scrambler>
  </b8b10>
</codingConfigurations>
```
Recommended Programming Techniques

**PCle 1/2**

```xml
<codingConfigurations>
  <b8b10>
    <fillerPrimitive symbol1="K28.5" symbol2="K28.0"/>
    <fillerPrimitive symbol1="K28.0"/>
    <scrambler>
      <resetPrimitive symbol1="K28.5"/>
    </scrambler>
  </b8b10>
</codingConfigurations>
```

**SATA**

```xml
<codingConfigurations>
  <b8b10>
    <fillerPrimitive symbol1="K28.5" symbol2="D10.2" symbol3="D10.2" symbol4="D27.3"/>
    <fillerPrimitive symbol1="K28.0"/>
    <scrambler polynomial="SATA" resetValue="61686" resetValueAfterHold="30349">
      <resetPrimitive symbol1="K28.3" symbol2="D21.5" symbol3="D23.1" symbol4="D23.1"/>
      <holdPrimitive symbol1="K28.3" symbol2="***" symbol3="***" symbol4="***"/>
      <pauseStartPrimitive symbol1="K28.3" symbol2="D20.5" symbol3="D25.4" symbol4="D25.4"/>
      <pauseEndPrimitive symbol1="K28.3" symbol2="***" symbol3="***" symbol4="***"/>
    </scrambler>
  </b8b10>
</codingConfigurations>
```
MIPI Phy

<codingConfigurations>
<b8b10>
  <fillerPrimitive symbol1="K28.1"/>
</b8b10>
</codingConfigurations>

SAS

<codingConfigurations>
<b8b10>
  <fillerPrimitive symbol1="K28.5" symbol2="D10.2" symbol3="D10.2" symbol4="D27.3"/>
  <fillerPrimitive symbol1="K28.5" symbol2="D7.0" symbol3="D7.0" symbol4="D7.0"/>
  <fillerPrimitive symbol1="K28.5" symbol2="D1.3" symbol3="D1.3" symbol4="D1.3"/>
  <fillerPrimitive symbol1="K28.5" symbol2="D27.3" symbol3="D27.3" symbol4="D27.3"/>
</b8b10>
</codingConfigurations>
Link Training Configuration

Link Training Configuration describes the parameters controlling the LTSSM (Link Training and Status State Machine). In the first release it is only defined and implemented for PCI Express 3.0. This is done with the <linkTrainingConfigurations> node. It is inserted just before the <codingConfigurations> node. The following parameters can be configured as attributes.

Table 18  Link Training Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>Possible Values</th>
<th>Default Value</th>
<th>Description</th>
<th>Used for DUT Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUT Type</td>
<td>AddInCard, SystemBoard</td>
<td>AddInCard</td>
<td>Specifies which role the DUT should play during link training. It can either be an upstream device for testing a downstream port or vice versa.</td>
<td>Both</td>
</tr>
<tr>
<td>Link Equalization</td>
<td>Bypass, PresetsOnly, Full</td>
<td>Bypass</td>
<td>Determines whether link equalization should be performed. It can either be aborted after phase 1 (Bypass) or fully executed. In the second case it can be determined whether only preset or all (that is, individual cursor) requests should be accepted.</td>
<td>Both</td>
</tr>
<tr>
<td>Lane Number</td>
<td>0 - 15</td>
<td>0</td>
<td>This is the lane number being used.</td>
<td>Both</td>
</tr>
<tr>
<td>Link Number</td>
<td>0 - 255</td>
<td>0</td>
<td>This is the link number being used.</td>
<td>Add-in Card</td>
</tr>
<tr>
<td>Start Preset</td>
<td>P0 - P9</td>
<td>P4</td>
<td>This is the preset used by the BERT's TX port after switching to Gen 3 operation and when operating as an upstream device.</td>
<td>Add-in Card</td>
</tr>
</tbody>
</table>
The following is an example of a Link Training Configuration for PCIe 3.0:

```xml
<linkTrainingConfigurations>
  <pcie3 dut="AddInCard" lane="0" link="0"
    linkEQ="PresetsOnly" generatorStartPreset="P4"
    dutRxPresetHint="Reserved" dutTxInitialPreset="P0"
    dutTxTargetPreset="P7" calibratedPresets="true" />
</linkTrainingConfigurations>
```
Recommended Programming Techniques

Downloading Pattern Sequence to Hardware

Once a sequence has been configured, it is downloaded to the hardware using the download button in the Sequence Editor shown in Figure 19 on page -60. When the sequence has been downloaded, this button turns green.

![Figure 19 Download button](image)

You can also download the pattern sequence configuration by copying and pasting the XML elements as a sequence string using the :DATA:SEQUence:VALue SCPI command. For an example, refer to :DATA:SEQUence[:SET]:VALue[?] on page 403.
Keysight M8000 Series of BER Test Solutions
Programming Guide

3 Programming Examples

- Introduction / 62
- Initializing the Connection / 62
- SJ Example / 64
- PJ Example / 70
- Sampling Point Alignment Example / 76
- BER Example / 80
- M8070B Jitter Tolerance Measurement CSV Output / 85
Introduction

This chapter provides programming examples for setting up some of the most common functions including:

- Initialization
- SJ
- PJ
- Sampling point alignment
- BER

If you have programmed the N4903B, examples are provided for comparison to the M8020A. This will facilitate code conversion from the N4903B to the M8020A.

Initializing the Connection

N4903B Initialization

The ResourceName ("TCPIP1::10.0.0.207::inst0::INSTR") must be replaced by the instrument's address string from the VISA Assistant.

' First our declarations...
Private myN490X As AgilentN490x
Private myBERT As AgilentBert
Private myPG As AgilentBertLib.IAgilentBertPG
Private myPGClock As AgilentBertLib.IAgilentBertPGClock
Private myPGOut As AgilentBertLib.IAgilentBertPGOutput
Private myEDDataIn As AgilentBertLib.IAgilentBertEDDataIn
Private Sub Form_Load()
  Set myN490X = New AgilentN490x
  Set myBERT = myN490x.IAgilentBert
  myBERT.Initialize ("TCPIP1::10.0.0.207::inst0::INSTR", True, True, "QueryInstrStatus=true")
End Sub
Private Sub Form_Unload(Cancel As Integer)
  myBERT.Close
End Sub
namespace baseInstrument.instruments
{
    class M8020A : BERT
    {
        private string m_PG_channel = "M1.DataOut1";
        private string m_ED_channel = "M1.DataIn1";
        private string m_system = "M1.ClkGen";
        double m_ED_clk = 16.0e9;
        double m_PG_clk = 16.0e9;
        double m_prevBER = 0.0;
        double m_prevBitCount = 0.0;
        double m_prevErrorCount = 0.0;
        double m_prevTimeStamp = 0.0;

        public M8020A(string resourceName,
                       string module = "M1",
                       string channel = "1") : base(resourceName)
        {
            if (module.IndexOf("MUX") == -1)
            {
                m_system = "M1.ClkGen";
                m_PG_channel = "M1.DataOut" + channel + "1";
                m_ED_channel = "M1.DataIn" + channel + "1";
            }
            else
            {
                m_system = "M1.ClkGen";
                m_PG_channel = "MUX";
                m_ED_channel = "MUX";
            }
        }
    }
}
SJ Example

N4903B SJ Example

```csharp
#region SJ
public override bool getSJEnabled()
// SOUR8:SIN?
{
    if ((int.Parse(this.Query("SOUR8:SIN?"))) == 1)
        return true;
    else
        return false;
}

public override double getSJFreq()
// SOUR8:SIN:FREQ?
{
    return double.Parse(this.Query("SOUR8:SIN:FREQ?"));
}

public override double getSJFreqMax()
// SOUR8:SIN:FREQ? MAX
{
    return double.Parse(this.Query("SOUR8:SIN:FREQ? MAX"));
}

public override double getSJFreqMin()
// SOUR8:PER:FREQ? MIN
{
    return double.Parse(this.Query("SOUR8:SIN:FREQ? MIN"));
}

public override List<string> setSJEnabled(bool b)
// SOUR8:SIN 1 ==> ON
// SOUR8:SIN 0 ==> OFF
{
    string strValue = "";
    if (b) strValue = "1"; 
    else strValue = "0";
    return this.Send("SOUR8:SIN: * + strValue");
}
```
public override List<string> setSJFreq(double SJFreq)
// SOUR8:SIN:FREQ
{
    string strValue = "";
    if ((this.getSJFreqMax() > SJFreq) && (this.getSJFreqMin() <= SJFreq))
        strValue = SJFreq.ToString();
    else
        strValue = this.getSJFreq().ToString();
    return this.Send(":=SOUR8:SIN:FREQ " + strValue);
}

public override double getSJAmp()
// SOUR8:SIN:LEV?
{
    return double.Parse(this.Query(":=SOUR8:SIN:LEV?"));
}

public override double getSJAmpMax()
// SOUR8:SIN:LEV? MAX
{
    return double.Parse(this.Query(":=SOUR8:SIN:LEV? MAX"));
}

public override double getSJAmpMin()
// SOUR8:SIN:LEV? MIN
{
    return double.Parse(this.Query(":=SOUR8:SIN:LEV? MIN"));
}

public override List<string> setSJAmp(double SJAmp)
// SOUR8:SIN:LEV
{
    string strValue = "";
    if ((this.getSJAmpMax() > SJAmp) && (this.getSJAmpMin() <= SJAmp))
        strValue = SJAmp.ToString();
    else
        strValue = this.getSJAmp().ToString();
    return this.Send(":=SOUR8:SIN:LEV " + strValue);
}
M8020A SJ Example

    #region SJ - implemented for M8020A
    public override bool getSJEnabled()
        // M8020A
        {
            string str = "";
            str = this.Query(":*SOUR:JITT:LFR:PER:STAT? " + m_PG_channel);

            if (str.IndexOf("1") != -1)
                return true;
            else
                return false;
        }

    public override double getSJFreq()
        // M8020A
        {
            string retString = "";
            double val = 0.0;

            retString = this.Query(":*SOUR:JITT:LFR:PER:FREQ? " + m_PG_channel);
            val = double.Parse(retString);
            return val;
        }

    public override double getSJFreqMax()
        // M8020A
        {
            string retString = "";
            double val = 0.0;

            retString = this.Query(":*SOUR:JITT:LFR:PER:FREQ? *,MAX" + m_PG_channel + ".");
            val = double.Parse(retString);
            return val;
        }
public override double getSJFreqMin()
// M8020A
{
    string retString = "";
    double val = 0.0;

    retString = this.Query(":SOUR:JITT:LFR:PER:FREQ? " + m_PG_channel + ",MIN");
    val = double.Parse(retString);
    return val;
}

public override List<string> setSJEnabled(bool b)
// M8020A
{
    string strValue = "";
    if (b) strValue = "ON";
    else strValue = "OFF";

    return this.Send(":SOUR:JITT:LFR:PER:STAT " + m_PG_channel + "," + strValue);
}

public override List<string> setSJFreq(double SJFreq)
// M8020A
{
    string strValue = "";
    SJFreq = Math.Round(SJFreq, 3);
    if ((this.getSJFreqMax() > SJFreq) && (this.getSJFreqMin() <= SJFreq))
        strValue = SJFreq.ToString();
    else if (this.getSJFreqMax() < SJFreq)
        strValue = this.getSJFreqMax().ToString();
    else if (this.getSJFreqMin() > SJFreq)
        strValue = this.getSJFreqMin().ToString();
    else
        strValue = this.getSJFreq().ToString();

    return this.Send(":SOUR:JITT:LFR:PER:FREQ " + m_PG_channel + "," + strValue +"Hz");
}
public override double getSJAmplitude()
// M8020A
{
    string retString = "";
    string unit = "";
    double val = 0.0;

    unit = this.Query(":\:SOUR:JITT:LFR:UNIT? * + m_PG_channel");
    this.Send(":\:SOUR:JITT:LFR:UNIT * + m_PG_channel + ",UINT");
    retString = this.Query(":\:SOUR:JITT:LFR:PER:AMPL? * + m_PG_channel");
    this.Send(":\:SOUR:JITT:LFR:UNIT * + m_PG_channel + ",* + unit");
    val = double.Parse(retString);
    return val;
}

public override double getSJAmplitudeMax()
// M8020A
{
    string retString = "";
    string unit = "";
    double val = 0.0;

    unit = this.Query(":\:SOUR:JITT:LFR:UNIT? * + m_PG_channel");
    this.Send(":\:SOUR:JITT:LFR:UNIT * + m_PG_channel + ",UINT");
    retString = this.Query(":\:SOUR:JITT:LFR:PER:AMPL? * + m_PG_channel + ",MAX");
    this.Send(":\:SOUR:JITT:LFR:UNIT * + m_PG_channel + ",* + unit");
    val = double.Parse(retString);
    return val;
}
public override double getSJAmpMin()
// M8020A
{
    string retString = "";
    string unit = "";
    double val = 0.0;

    unit = this.Query("*:SOUR:JITT:LFR:UNIT? " + m_PG_channel);
    this.Send("*:SOUR:JITT:LFR:UNIT " + m_PG_channel + ",UINT");
    retString = this.Query("*:SOUR:JITT:LFR:PER:AMPL? " + m_PG_channel + ",MIN");
    this.Send("*:SOUR:JITT:LFR:UNIT " + m_PG_channel + "," + unit);
    val = double.Parse(retString);
    return val;
}

public override List<string> setSJAmp(double SJAm)
// M8020A
{
    string strValue = "";
    string unit = "";
    double amp = 0.0;

    if (m_PG_channel == "MUX")
        SJAm /= 2.0;
    if ((this.getSJAmpMax() > SJAm) && (this.getSJAmpMin() <= SJAm))
        amp = SJAm;
    else
        amp = this.getSJAmp();

    amp = Math.Round(amp, 3);
    strValue = amp.ToString();
    unit = this.Query("*:SOUR:JITT:LFR:UNIT? " + m_PG_channel);
    this.Send("*:SOUR:JITT:LFR:UNIT " + m_PG_channel + ",UINT");
    this.Send("*:SOUR:JITT:LFR:PER:AMPL " + m_PG_channel + "," + strValue);
    return this.Send("*:SOUR:JITT:LFR:UNIT " + m_PG_channel + "," + unit);
}
PJ Example

N4903B PJ Example

```csharp
#region PJ
public override bool getPJEnabled()
// SOUR8:PER?
{
    if ((int.Parse(this.Query("*:SOUR8:PER?"))) == 1)
        return true;
    else
        return false;
}

public override double getPJFreq()
// SOUR8:PER:FREQ?
{
    return double.Parse(this.Query("*:SOUR8:PER:FREQ?"));
}

public override double getPJFreqMax()
// SOUR8:PER:FREQ? MIN MAX
{
    return double.Parse(this.Query("*:SOUR8:PER:FREQ? MAX"));
}

public override double getPJFreqMin()
// SOUR8:PER:FREQ? MIN MAX
{
    return double.Parse(this.Query("*:SOUR8:PER:FREQ? MIN"));
}

public override List<string> setPJEnabled(bool b)
// SOUR8:PER 1 ==> ON
// SOUR8:PER 0 ==> OFF
{
    string strValue = "";
    if (b) strValue = "1";
    else strValue = "0";
    return this.Send("*:SOUR8:PER: " + strValue);
}
```
public override List<string> setPJFreq(double PJFreq)
// SOUR8:PER:FREQ
{
    string strValue = "";
    if ((this.getPJFreqMax() > PJFreq) && (this.getPJFreqMin() <= PJFreq))
        strValue = PJFreq.ToString();
    else if (this.getPJFreqMax() < PJFreq)
        strValue = this.getPJFreqMax().ToString();
    else if (this.getPJFreqMin() > PJFreq)
        strValue = this.getPJFreqMin().ToString();
    else
        strValue = this.getPJFreq().ToString();
    return this.Send("\":SOUR8:PER:FREQ " + strValue);
}

public override double getPJAmp()
// SOUR8:PER:LEV?
{
    return double.Parse(this.Query("\":SOUR8:PER:LEV?\""));
}

public override double getPJAmpMax()
// SOUR8:PER:LEV? MAX
{
    return double.Parse(this.Query("\":SOUR8:PER:LEV? MAX\""));
}

public override double getPJAmpMin()
// SOUR8:PER:LEV? MIN
{
    return double.Parse(this.Query("\":SOUR8:PER:LEV? MIN\""));
}

public override List<string> setPJAmp(double PJAmp)
// SOUR8:PER:LEV
{
    string strValue = "";
    if ((this.getPJAmpMax() > PJAmp) && (this.getPJAmpMin() <= PJAmp))
        strValue = PJAmp.ToString();
    else
        strValue = this.getPJAmp().ToString();
    return this.Send("\":SOUR8:PER:LEV " + strValue);
}
M8020A PJ Example

```csharp
#region PJ - as PJ1 implemented for M8020A
public override bool getPJEnabled()
// M80820A - checked
{
    string str = "";
    str = this.Query(":\SOUR:JITT:HFR:PER1:STAT? " + m_PG_channel);
    if (str.IndexOf("1") != -1)
        return true;
    else
        return false;
}

public override double getPJFreq()
// M8020A - checked
{
    string retString = "";
    double val = 0.0;
    retString = this.Query(":\SOUR:JITT:HFR:PER1:FREQ? " + m_PG_channel);
    val = double.Parse(retString);
    return val;
}

public override double getPJFreqMax()
// M8020A - checked
{
    string retString = "";
    double val = 0.0;
    retString = this.Query(":\SOUR:JITT:HFR:PER1:FREQ? " + m_PG_channel + " MAX");
    val = double.Parse(retString);
    return val;
}
#endregion
```
public override double getPJFreqMin()
    // M8020A - checked
    {
        string retString = "";
        double val = 0.0;
        retString = this.Query("" + m_PG_channel + ",MIN")
            .Trim();
        val = double.Parse(retString);
        return val;
    }

public override List<string> setPJEnabled(bool b)
    // M8020A - checked
    {
        string strValue = "";
        if (b) strValue = "ON";
        else strValue = "OFF";
        return this.Send("" + m_PG_channel + "," + strValue);
    }

public override List<string> setPJFreq(double PJFreq)
    // M8020A - checked
    {
        string strValue = "";
        PJFreq = Math.Round(PJFreq, 3);
        if ((this.getPJFreqMax() > PJFreq) && (this.getPJFreqMin() <= PJFreq))
            strValue = PJFreq.ToString();
        else if (this.getPJFreqMax() < PJFreq)
            strValue = this.getPJFreqMax().ToString();
        else if (this.getPJFreqMin() > PJFreq)
            strValue = this.getPJFreqMin().ToString();
        else
            strValue = this.getPJFreq().ToString();
        return this.Send("" + m_PG_channel + "," + strValue + "Hz");
    }
public override double getPJAmplitude()
// M8020A - checked
{
    string retString = "*; string unit = "*;
    double val = 0.0;
    unit = this.Query("*:SOUR:JITT:HFR:UNIT\? * + m_PG_channel");
    this.Send("*:SOUR:JITT:HFR:UNIT * + m_PG_channel + ",UINT");
    retString = this.Query("*:SOUR:JITT:HFR:PER1:AMPL\? * + m_PG_channel");
    this.Send("*:SOUR:JITT:HFR:UNIT * + m_PG_channel + ",unit");
    val = double.Parse(retString);
    return val;
} 

public override double getPJAmplitudeMax()
// M8020A - checked
{
    string retString = "*; string unit = "*;
    double val = 0.0;
    unit = this.Query("*:SOUR:JITT:HFR:UNIT\? * + m_PG_channel");
    this.Send("*:SOUR:JITT:HFR:UNIT * + m_PG_channel + ",UINT");
    retString = this.Query("*:SOUR:JITT:HFR:PER1:AMPL\? * + m_PG_channel + ",MAX");
    this.Send("*:SOUR:JITT:HFR:UNIT * + m_PG_channel + ",unit");
    val = double.Parse(retString);
    return val;
} 

public override double getPJAmplitudeMin()
// M8020A - checked
{
    string retString = "*; string unit = "*;
    double val = 0.0;
    unit = this.Query("*:SOUR:JITT:HFR:UNIT\? * + m_PG_channel");
    this.Send("*:SOUR:JITT:HFR:UNIT * + m_PG_channel + ",UINT");
    retString = this.Query("*:SOUR:JITT:HFR:PER1:AMPL\? * + m_PG_channel + ",MIN");
    this.Send("*:SOUR:JITT:HFR:UNIT * + m_PG_channel + ",unit");
    val = double.Parse(retString);
    return val;
}
public override List<string> setPJAmp(double PJ Amp) {
    string strValue = "";
    string unit = "";
    double amp = 0.0;

    if (m_PG_channel == "MUX")
        PJ Amp /= 2.0;

    if ((this.getPJAmpMax() > PJ Amp) && (this.getPJAmpMin() <= PJ Amp))
        amp = PJ Amp;
    else
        amp = this.getPJAmp();

    amp = Math.Round(amp, 3);
    strValue = amp.ToString();

    unit = this.Query(":\SOU\:JITT\:HFR\:UNIT? " + m_PG_channel);
    this.Send(":\SOU\:JITT\:HFR\:UNIT " + m_PG_channel + ",UINT");
    this.Send(":\SOU\:JITT\:HFR\:PER1\:AMPL " + m_PG_channel + "," + strValue);

    return this.Send(":\SOU\:JITT\:HFR\:UNIT " + m_PG_channel + "," + unit);
}
Sampling Point Alignment Example

N4903B Sampling Point Alignment Example

    public bool checkN900()
    {
        return m_N4900;
    }

# region Sampling Point Alignment

public override string autoAlign()
{
    string result = "";
    this.Send("*:SENS1:EYE:ALIGN:AUTO 1");
    this.opc();
    do
    {
        result = this.Query("*:SENS1:EYE:ALIGN:AUTO?");
    } while ((result != "CS_ABORTED") && (result != "CS_FAILED") && (result != "CS_SUCCESSFUL");
    return result;
}

public override string dataCenter()
{
    string result = "";
    this.Send("*:SENS1:EYE:TCEN 1");
    this.opc();
    do
    {
        result = this.Query("*:SENS1:EYE:ALIGN:AUTO?");
    } while ((result != "CS_ABORTED") && (result != "CS_FAILED") && (result != "CS_SUCCESSFUL"));
    return result;
}

public override string thresholdCenter()
{
    string result = "";
    this.Send("*:SENS1:EYE:ACEN 1");
    this.opc();
    do
    {
        result = this.Query("*:SENS1:EYE:ALIGN:AUTO?");
    } while ((result == "CS_ABORTED") || (result == "CS_FAILED") || (result == "CS_SUCCESSFUL"));
    return result;
}
public override List<string> sync()
{
    return this.Send("*:SENS1:SYNC 1");
}

#region sampling point
public override List<string> setSamplingPointDelay(double x)
{
    return this.Send("*:INP1:DEL * + x.ToString()" );
}

public override double getSamplingPointDelay()
{
    string str = "";
    double delay = 0.0;

    str = this.Query("*:INP1:DEL?" );
    delay = double.Parse(str);
    return delay;
}

public override List<string> setSamplingPointDelayRel(double x)
{
    double current = 0.0;

    current = this.getSamplingPointDelay();
    x += current;

    return setSamplingPointDelay(x);
}
#endregion
M8020A Sampling Point Alignment Example

```csharp
public override string autoAlign()
{
    string result = "";

    /*
    * do
    { 
        result = this.Query("*:STATus:OPERation:RUN:EVENT?"+m_ED_channel);
    } while ((int.Parse(result) % 2) == 0);
    */

    this.Send("*:INP:ALIG:EYE:AUTO "+ m_ED_channel);
    this.opc();

    /*
    do 
    { 
        result = this.Query("*:INP:ALIG:EYE:RES:DEL? "+ m_ED_channel);
    } while (result == "");
    */
    return result;
}

dataCenter()
{
    string result = "";

    this.Send("*:INP:ALIG:EYE:TCEN "+ m_ED_channel);
    this.opc();

    /*
    do
    { 
        result = this.Query("*:INP:ALIG:EYE:RES:DEL? "+ m_ED_channel);
    } while (result == "");
    */
    return result;
}
```
public override string thresholdCenter()
{
    string result = "";

    this.Send("":INP:ALIG:EYE:ACEN " + m_ED_channel);
    this.opc();
    /*
    do
    { 
        result = this.Query("":INP:ALIG:EYE:RES:THR? " + m_ED_channel);
    } while (result == "");
    */
    return result;
}

public override List<string> sync()
{
    return this.Send("":DATA:SYNC " + m_ED_channel);
}

#endregion

#region sampling point - implemented for M8020A

public override List<string> setSamplingPointDelay(double x)
{
    string delay = "";
    delay = (1.0e9 * x).ToString() + "e-9";
    return this.Send("":INP:DEL " + m_ED_channel + "," + x.ToString());
}

public override double getSamplingPointDelay()
{
    string retString = "";
    double val = 0.0;
    retString = this.Query("":INP:DEL? " + m_ED_channel);
    val = double.Parse(retString);
    return val;
}
#endregion
3

Programming Examples

BER Example
N4903B BER Example

# region ber
private double m_N4900_BitCount = 0.0;
private double m_N4900_ErrorCount = 0.0;
public override List<string> resetBERCounter()
{
m_N4900_BitCount = 0.0;
m_N4900_ErrorCount = 0.0;
return this.startBERAcc();
}
public override List<string> setBERCounterResults()
{
this.stopBERAcc();
this.startBERAcc();
double errors = 0.0;
double bits = 0.0;
errors = double.Parse(this.Query(":PFETCH:SENS2:ECO?"));
bits = double.Parse(this.Query(":PFETCH:SENS2:BCO?"));
m_ErrorCount = m_N4900_ErrorCount + errors;
m_N4900_ErrorCount = m_ErrorCount;
m_BitCount = m_N4900_BitCount + bits;
m_N4900_BitCount = m_BitCount;
m_BER = m_ErrorCount / m_BitCount;

}

return this.ErrorQ();

public override List<string> startBERAcc()
{
return this.Send(":SENS1:GATE ON");
}
public override List<string> stopBERAcc()
{
return this.Send(":SENS1:GATE OFF");
}
public override double getInstantaneousBER()
{
return double.Parse(this.Query(":FETC:SENS2:ERAT?"));
}
# endregion

80

Keysight M8000 Series of BER Test Solutions Programming Guide


M8020A BER Example

```csharp
private double m_prevAccBER_M8020 = 0.0;
private double m_prevAccBitCount_M8020 = 0.0;
private double m_prevAccErrorCount_M8020 = 0.0;

public override List<string> startBERAcc()
// M8020A
{
    List<string> errorStr = new List<string>();
    errorStr = this.ErrorQ();
    return errorStr;
}

public override List<string> stopBERAcc()
// M8020A - error counters can't be stopped yet
{
    List<string> errorStr = new List<string>();
    errorStr = this.ErrorQ();
    return errorStr;
}

public override List<string> resetBERCounter()
{
    List<string> errorStr = new List<string>();
    List<double> bitResults = new List<double>();
    List<double> bitCounter = new List<double>();
    bitResults = this.getErrorCounter();
    errorStr = this.ErrorQ();
    m_prevAccBitCount_M8020 = bitResults[0];
    m_prevAccErrorCount_M8020 = bitResults[1];
    m_prevAccBER_M8020 = m_prevAccErrorCount_M8020 / m_prevAccBitCount_M8020;
    bitCounter = this.getErrorCounter();
    m_prevBitCount = bitCounter[0];
    m_prevErrorCount = bitCounter[1];
    m_prevTimeStamp = bitCounter[2];
    return errorStr;
}
```
public override double getInstantaneousBER()
  // M8020A
{
  double accumulatedBER = 0.0;
  List<double> result = new List<double>();
  result = this.getBERAccResults();
  accumulatedBER = result[2];
  return accumulatedBER;
}

public override List<string> setBERCounterResults()
  // M8020A
{
  List<double> berResults = new List<double>();
  List<double> bitCounter = new List<double>();
  double bitCount = 0.0;
  double errorCount = 0.0;
  double elapsedTime = 0.0;
  double timeStamp = 0.0;
  double ber = 0.5;
  
  bitCounter = this.getErrorCounter();
  bitCount = bitCounter[0] - m_prevBitCount;
  errorCount = bitCounter[1] - m_prevErrorCount;
  timeStamp = bitCounter[2];
  elapsedTime = 1000.0 * (timeStamp - m_prevTimeStamp);
  ber = errorCount / bitCount;
  
  m_BitCount = bitCount;
  m_ErrorCount = errorCount;
  m_BER = ber;

  return this.ErrorQ();
}

private List<double> getErrorCounter()
  // M8020A
{
  List<double> bitCounter = new List<double>();
  double bitCount = 0.0;
  double errorCount = 0.0;
  double timeStamp = 0.0;
  double counted1s = 0.0;
  double counted0s = 0.0;
  double erroneous1s = 0.0;
  double erroneous0s = 0.0;
  string identifier = "";
  string resultStr = "",
  string valStr = "",
  int index = 0;
resultStr = this.Query("^FETC:BCO? " + m_ED_channel);

// timestamp
index = resultStr.IndexOf(";", index - 1);
timeStamp = double.Parse(valStr);
resultStr = resultStr.Substring(index + 1, resultStr.Length - index - 1);

// identifier
index = resultStr.IndexOf(";", index - 1);
valStr = resultStr.Substring(0, index);
identifier = valStr;
resultStr = resultStr.Substring(index + 1, resultStr.Length - index - 1);

// counted 1s
index = resultStr.IndexOf(";", index - 1);
valStr = resultStr.Substring(0, index - 1);
counted1s = double.Parse(valStr);
resultStr = resultStr.Substring(index + 1, resultStr.Length - index - 1);

// counted 0s
index = resultStr.IndexOf(";", index - 1);
valStr = resultStr.Substring(0, index - 1);
counted0s = double.Parse(valStr);
resultStr = resultStr.Substring(index + 1, resultStr.Length - index - 1);

// erroneous 1s
index = resultStr.IndexOf(";", index - 1);
valStr = resultStr.Substring(0, index - 1);
erroneous1s = double.Parse(valStr);
resultStr = resultStr.Substring(index + 1, resultStr.Length - index - 1);

// erroneous 0s
index = resultStr.IndexOf(";", index - 1);
valStr = resultStr.Substring(0, index - 1);
erroneous0s = double.Parse(valStr);
resultStr = resultStr.Substring(index + 1, resultStr.Length - index - 1);

bitCount = counted0s + counted1s;
//bitCount -= m_prevBitCount;
errorCount = erroneous0s + erroneous1s;
//errorCount -= m_prevErrorCount;

bitCounter.Add(bitCount);
bitCounter.Add(errorCount);
bitCounter.Add(timeStamp);
return bitCounter;
}

# endregion
If the M8070B software is running on Windows 8 or 8.1, the VXI-11 protocol is not supported.

Example:

“SCPI Access (VXI-11): TCPIPO::localhost::inst0::INSTR”

does not work on Windows 8 or 8.1.

In this case, the HiSLIP protocol should be used as shown below:

“SCPI Access (HiSLIP): TCPIPO::localhost::hislip0::INSTR”
M8070B Jitter Tolerance Measurement CSV Output

The following python script generates the results from all open jitter tolerance measurements to different CSV files. This script can be easily modified to generate the output of the highest-passing point at each jitter modulation frequency value measured. This script can be run from the M8070B Script Editor or can also be run externally.

```python
# M8020A jtol CSV output
#
# Run from the M8070B script editor

def WaitForCompletion():
    res = M8000.Scpi.Query(':syst:err?')
    if (res != '0","No error")':
        print res

WaitForCompletion()
M8000.Scpi.Timeout = 120000

# loop through all jitter tolerance measurements
for id in M8000.Scpi.Query(':PLUG:JTOL:CAT?').Split(',
    print "Measurement: " + id + ":"

    # print results
    # size of results
    print " " + str(len(results)) + " data items, " + str(int(len(results)/6)) + " rows;",

    # if there are any rows (and not just a blank measurement), then
    # use the name of the jtol measurement as the filename
    # this will overwrite the file if it already exists!
    # the file will be located in \\Program Files (x86)\\Keysight\\M8070B\\bin", mayb
    if (int(len(results)/6) == 0):
        # nothing to write to file
        print "no file created"
    else:
        with open(id.Trim(‘’).Replace(‘ ,’ + ".csv", ‘w’) as f:
            # output header to file
            f.write("Modulation Frequency (GHz),Sinusoidal Jitter (UI),Number of E Ratio,Measurement results\n")
```

```python
# output header to file
f.write("Modulation Frequency (GHz),Sinusoidal Jitter (UI),Number of E Ratio,Measurement results\n")
```
# iterate over the data
# first element ([0]) is a module identifier, discard
# items are then grouped in bundles of six, per header above
# we need to insert a comma or LF after each item, depending on pos
for idx, data in enumerate(results):
    if (idx==0):
        # do nothing, this is the module identifier
        pass
    elif (idx%6==0):
        # output ending in a newline
        # print data
        f.write(data + "\n")
    else:
        # output ending in a comma
        # print data + ","
        f.write(data + ",")

# close file and move on!
f.close()
print "written to file"
4 SCPI Command Language
SCPI Command Language - Introduction

The Serial BERT is compatible with the standard language for remote control of instruments. Standard Commands for Programmable Instruments (SCPI) is the universal programming language for instrument control.

SCPI can be subdivided into the following command sets:

- SCPI Common Commands
- SCPI Instrument Control Commands
- IEEE 488.2 Mandatory Commands

SCPI Common Commands

This is a common command set. It is compatible with IEEE 488.2 and contains general housekeeping commands. The common commands are always preceded by an asterisk. A typical example is the reset command:

*RST

The IEEE 488.2 command set also contains query commands. Query commands always end with a question mark.

SCPI Instrument Control Commands

The programming commands are compatible with the Standard Commands for Programmable Instruments (SCPI) standard. For more detailed information regarding the GPIB, the IEEE 488.2 standard, or the SCPI standard, refer to the following books:

IEEE 488.2 Mandatory Commands

In order to comply with the SCPI model as described in IEEE 488.2, the Serial BERT implements certain mandatory commands. Other commands are implemented optionally. For more detail on the IEEE 488.2 mandatory and optional commands, see IEEE Commands - Reference on page 101.

Overlapped and Sequential Commands

IEEE 488.2 defines the distinction between overlapped and sequential commands. A sequential command is one which finishes executing before the next command starts executing. An overlapped command is one which does not finish executing before the next command starts executing.

**NOTE**

It is not reliable to use wait statements in the control program to facilitate the use of overlapped commands.

Because these commands may allow the execution of more than one command at a time, special programming techniques must be used to ensure valid results. The common commands *OPC, *WAI and *OPC? can be used for this purpose. They help synchronize a device controller with the execution of overlapped commands.

The behaviors of these commands, in brief, are as follows:

- **OPC**
  The *OPC command sets the Operation Complete (OPC) bit of the Event Register when the No Operation Pending flag is TRUE (No Operation Pending flag is attached to each overlapped command). Until that time, the controller may continue to parse and execute previous commands. It is good technique, then, to periodically poll the OPC bit to determine if the overlapped command has completed.

- **WAI**
  The *WAI command allows no further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.
  The *WAI command can be used for overlapped commands. It stops the program execution until any pending overlapped commands have finished. Specifically, it waits until the No Operation Pending flag is TRUE, or receipt of a dcas message, or a power on.
• *OPC?
  The *OPC? query returns the ASCII character "1" in the Output Queue when the No Operation Pending flag is TRUE. At the same time, it also sets the Message Available (MAV) bit in the Status Byte Register. The *OPC? will not allow further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.

NOTE
The command behaviors described above are for overlapped commands. When the same commands are used with sequential commands, the behaviors may be different.

Data Types

The M8020A/M8030A/M8040A has the capability of receiving and returning data in the following formats:

STRING
A string of human-readable ASCII characters, either quoted or nonquoted.

NUMERIC
The M8020A/M8030A/M8040A handles the following numeric formats:

- <NR1>: Integer (0, 1, 2, -1, etc.)
- <NR2>: Number with an embedded decimal point (0.1, 0.001, 3.3, etc.)
- <NR3>: Number with an embedded decimal point and exponent (1e33, 1.3e-12, etc.)
- <NRf>: Represents <NR1>, <NR2> and <NR3>
- Binary preceded by #b (#b010101, #b011111, etc.)
- Octal preceded by #q (#Q777111, #q7331777, etc.)
- Hex preceded by #h (#haff, #h8989fffff, etc.)

BOOLEAN
Boolean values can be sent to the M8020A/M8030A/M8040A as either ON | OFF or 0 | 1. The M8020A/M8030A/M8040A answers queries with 0 | 1.
Definite Length Arbitrary Block Data

Block data is used when a large quantity of related data is being returned. A definite length block is suitable for sending blocks of 8-bit binary information when the length is known beforehand. An indefinite length block is suitable for sending blocks of 8-bit binary information when the length is not known beforehand or when computing the length beforehand is undesirable.

It has the following format:

```
#<Length of length><Length of data><data>
```

<Length of length> is a single integer that contains the number of digits in <Length of data>, which in turn contains the length of the data. For example, a 512-byte pattern would be defined as:

```
#3512<data>
```

Important Points about SCPI

Important Points about SCPI - Concepts

There are a number of key areas to consider when using SCPI for the first time. These are as follows:

- Instrument Model
- Command Syntax
- Optional Parts of Commands
- Sending Commands
- Command Separators
- SCPI Command Structure

Instrument Model

SCPI guidelines require that the M8020A/M8030A/M8040A is compatible with an instrument model. This ensures that when using SCPI, functional compatibility is achieved between instruments that perform the same tasks. For example, if two different instruments have a programmable clock frequency setting, then both instruments would use the same SCPI commands to set their frequency. The instrument model is made up of a number of subsystems.

The sub-system defines a group of functions within a module and has a unique identifier under SCPI, which is called the Root Keyword.
For more details on the instrument model, see STATus Subsystem on page 111.

Command Syntax

Commands may be up to twelve characters long. A short-form version is also available which has a preferred length of four characters or less. In this document the long-form and short-form versions are shown as a single word with the short-form being shown in upper-case letters.

For example, the long-form node command SOURce has the short-form SOUR. Using the short form saves time when entering a program; however, using the long form makes a program more descriptive and easier to understand.

SCPI commands may be commands only, commands and queries, or queries only. A question mark at the end of a command indicates that it is a query. If the question mark appears in brackets ([?]), the command has a command and query form.

Optional Command Keywords

Some layers in the SCPI command structure are optional. These optional keywords are indicated by square brackets ([ ]). A typical use for these types of keywords is with a command that is unique to one module. In this case, the top layer (Root Keyword) of the command structure may be omitted.

For example, the following command code segments are functionally identical:

[:SOURce]:JITTer[:GLOBa][:STATe] <ON|OFF|1|0>

:JITTer <ON|OFF|1|0>

:JITT <ON|OFF|1|0>

:jitt <ON|OFF|1|0>

Note that it is not necessary to include the syntax inside the square brackets ([ ]).
Query Responses

It is possible to interrogate the individual settings and status of a device using query commands. Retrieving data is a two-stage operation.

The query command is sent from the controller using the OUTPUT statement and the data is read from the device using the ENTER statement. A typical example is the SCPI IEEE 488.2 Common Command *IDN? which queries the identity of a device.

NOTE

When sending strings to the instrument, either the double quote (") or the single quote may be used ('), the latter being more suited to PASCAL programs, which make use of a single quote; the former being more suited to use in BASIC programs, which use a double quote as a delimiter.

Command Separators

The SCPI command structure is hierarchical and is governed by commas, semicolons and colons:

• Commas are used to separate parameters in one command.
• Colons are used to separate levels.
• Semicolons are used to send more than one command to the instrument at a time.

It is possible to send several commands in one pass, as long as the commands all belong to the same node in the SCPI tree. The commands have to be separated by semicolons.

The following SCPI commands provide examples of this.

SOURce:VOLTage:OFFSet 'M2.DataOut2',-0.99
SOURce:VOLTage:AMPLitude 'M2.DataOut2',1.11

These commands can also be sent as follows:

VOLT:OFFS 'M2.DataOut2','-0.99; 'M2.DataOut2',AMPL 1.11
SCPI Command Structure Example

The SCPI command structure can be best examined by means of an example. For example, the command to set the pattern generator’s output amplitude is:

`:SOURce:VOLTage[:AMPLitude] 'M1.DataOut1',1.11

The structure of this command can be illustrated as follows:

| [:SOURce] | This is the top layer of the command structure and identifies the source subsystem. |
| :VOLTage | This is the next layer and defines the subnode for setting a voltage level. |
| [:AMPLitude] | This is the command itself for setting the output amplitude level. |
| 'M1.DataOut1',1.11 | This specifies pattern generator 1, channel 1 and specifies an amplitude of 1.11. |

NOTE

Any optional commands are enclosed in square brackets [ ] and any optional characters are shown in lower case.

A colon indicates a change of level in the command hierarchy. Commands at the same level in the hierarchy may be included in the same command line, if separated by a semi-colon.

The bar symbol (|) indicates mutually exclusive commands.

To translate this syntax into a command line, follow the convention described above. Remember, however, that the command line can be created in several different ways. It can be created with or without optional keywords and in a long or short form. The following example gives three possible forms of the command line; all are acceptable.

In long form:

`:SOURcE:VOLTage:AMPLitude 'M1.DataOut1',1.11

In short form:

`:SOUR:VOLT:AMPL 'M1.DataOut1',1.11
With the optional commands removed:

:VOLT 'M1.DataOut1',1.11

The long form is the most descriptive form of programming commands in SCPI.

Sending Commands to the M8020A/M8030A/M8040A

A command is invalid and will be rejected if:

- It contains a syntax error.
- It cannot be identified.
- It has too few or too many parameters.
- A parameter is out of range.
- It is out of context.

Sending Commands using VISA

The following is a list of the available hardware interfaces for sending commands to the M8020A/M8030A/M8040A firmware:

SCPI Access (HiSLIP): TCPIP0::localhost::hislip0::INSTR (High-Speed LAN Instrument Protocol)

SCPI Access (VXI-11): TCPIP0::localhost::inst0::INSTR (VXI-11 is a TCP/IP instrument protocol defined by the VXIbus Consortium)

SCPI Access (Socket): TCPIP0::localhost::5025::SOCKET (Standard SCPI-over-sockets port)

SCPI Access (Telnet): telnet localhost 5024 (Communication with LAN instrument through SCPI Telnet port)

NOTE

If you use the VXI-11 (TCP/IP instrument protocol) in your test programs, you must change the resource string to the HiSLIP protocol if the software is running on Windows 8 or Windows 8.1. VXI-11 is not supported at this time on Windows 8 or Windows 8.1.

For example:

“TCPIP0::192.17.34.0::inst0::INSTR” -> “TCPIP0::192.17.34.0::hislip0::INSTR”
Command Line Arguments

(See Communication on page 97 for details about /Socket, /Telnet, /Inst, /AutoID, /NoAutoID, /FallBack).

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/AutoId</td>
<td>Start in auto ID mode (this is the default) [optional]</td>
</tr>
<tr>
<td>/NoAutoId</td>
<td>Do not start in auto ID mode - use communication parameters from command line [optional]</td>
</tr>
<tr>
<td>/FallBack</td>
<td>Use auto ID mode if the communication parameters from the command line don’t work [optional]</td>
</tr>
<tr>
<td>/Socket &lt;socket port&gt;</td>
<td>Set the socket port for the SCPI communication (only used with /NoAutoId) [optional]</td>
</tr>
<tr>
<td>/Telnet &lt;telnet port&gt;</td>
<td>Set the telnet port for the SCPI communication (only used with /NoAutoId) [optional]</td>
</tr>
<tr>
<td>/Inst &lt;instrument number&gt;</td>
<td>Set the instrument number for VXI-11.3 and HiSIP SCPI communication [optional]</td>
</tr>
<tr>
<td>/rcl</td>
<td>Recall last used setting [optional]</td>
</tr>
<tr>
<td>/rst</td>
<td>Reset to factory default [optional]</td>
</tr>
<tr>
<td>/IgnoreAwg</td>
<td>M8070B software don’t grab M8195A modules [optional]</td>
</tr>
</tbody>
</table>
Communication

Depending on the command line arguments /Socket, /Telnet, /Inst, /AutoID, /NoAutoID, /FallBack, the M8070B software starts several servers to handle SCPI commands. (Refer to the table above.)

/Socket, /Telnet, /Inst: If -1, don’t start the respective servers

*Defaults:
Socket port: 5025 (e.g. TCPIP0::localhost::5025::SOCKET)
Telnet port: 5024
HiSLIP, VXI-11.3: 0 (e.g. TCPIP0::localhost::hislip0::INSTR, TCPIP0::localhost::inst0::INSTR)

/FallBack: If starting a server fails because of a conflict, try using another port or number
*HiSLIP, VXI-11.3: increase the index until a server can be started successfully

*Socket, Telnet: start with port 60000, then increase it until the servers can be started successfully. If neither socket nor telnet is disabled the M8070B software tries to start the servers on two consecutive ports
  (socket port = telnet port + 1)

/AutoID: Automatically select ports and number for the connections, which are unique per instrument.
*This is the default behavior; it is not necessary to specify this argument on the command line.
*/Socket, /Telnet, /Inst are ignored (unless they are -1 and a server is disabled)

*If the M8070B software detects more than one AXIe module, use a special mechanism to obtain a number for the HiSLIP and VXI-11.3 servers, which makes sure that the M8070B software always uses the same VISA resource string per module
*The socket and telnet port are then calculated from the HiSLIP index:
  telnet port = 60000 + 2 * <HiSLIP index>
  socket port = 60000 + 2 * <HiSLIP index> + 1
Note: Ports may already be in use by Windows or other applications, so they are not available for M8195A.

/NoAutoID: Do not automatically select ports and number for the connections, use the values specified with /Socket, /Telnet, /Inst or their respective default values instead.

If both /NoAutoID and /AutoID are specified, /AutoID overrides /NoAutoID.

NOTE
The first port not assigned by IANA is 49152 (IANA, Internet Assigned Numbers Authority, http://www.iana.org).
5 SCPI Command Reference

Subsystems / 100
IEEE Commands - Reference / 101
SCPI Standard Commands / 107
Miscellaneous Commands / 108
Command Syntax to Find Min/Max Values / 109
Location and Module Mapping - Reference / 110
STATus Subsystem / 111
TRIGger Subsystem / 132
Source Subsystem / 139
OUTPut Subsystem / 211
EVENt Subsystem / 248
INPut Subsystem / 250
SENSe Subsystem / 298
CLOCk Subsystem / 299
N1000 Subsystem / 303
FETCH Subsystem / 312
MMEMory Subsystem / 317
ACQuisition subsystem / 323
PLUGin Subsystem / 328
SYSTem Subsystem / 372
DATA Subsystem / 391
Subsystems

The SCPI commands are divided into subsystems, which reflect the various functionality of the instrument. The following figure shows where the port-related subsystems are located.

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATus</td>
<td>The STATus subsystem reports the current system condition, when an event changes and sets transition filter states to report when a passing event occurs.</td>
</tr>
</tbody>
</table>
| TRIGger   | The TRIGger subsystem selects the following trigger modes:  
  - INTernal selects the 100 MHz reference clock of the AXIe chassis.  
  - REFerence selects a 10 MHz/100 MHz external reference clock.  
  - DIRect selects the clock in the M8041A and can be set manually from 8.1 GHz to 16.2 GHz as the system frequency.  
  - CMULTiplier selects a multiplied or divided clock to specify the system frequency. |
| SOURce    | The SOURce subsystem controls output signals (for example, for setting frequency and levels). |
| OUTPut    | The OUTPut subsystem controls the output ports of the pattern generator. |
| INPut     | The INPut subsystem controls the input ports of the error detector. |
| CLOCk     | The CLOCk subsystem is used to detect clock source or to set the clock frequency applied to input or output channel. |
| FETCH     | The FETCH subsystem is used to query the error detector’s results. |
| MMEMory   | The MMEMory subsystem is used to store and recall predefined or custom created settings and patterns. |
| ACQuisition | The acquisition subsystem controls how a real-time oscilloscope is being configured in order to capture the signal for BER measurements. |
| PLUGin    | The PLUGin subsystem is a plugin interface for integrating C# assemblies into the M8070B user interface and instrument software. |
| SYSTem    | The SYSTem subsystem is used for general system functions. |
| DATA      | The DATA subsystem is used to select patterns, define symbol parameters, define sequence blocks and loops and synchronize the pattern. |
IEEE Commands - Reference

Mandatory Commands

The following mandatory IEEE 488.2 commands are implemented:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CLS</td>
<td>*CLS on page 102</td>
</tr>
<tr>
<td>*CAL?</td>
<td>*CAL? on page 102</td>
</tr>
<tr>
<td>*LRN?</td>
<td>*LRN? on page 102</td>
</tr>
<tr>
<td>*ESE[?]</td>
<td>*ESE[?] on page 102</td>
</tr>
<tr>
<td>*ESR?</td>
<td>*ESR? on page 102</td>
</tr>
<tr>
<td>*IDN?</td>
<td>*IDN? on page 103</td>
</tr>
<tr>
<td>*OPC</td>
<td>*OPC on page 103</td>
</tr>
<tr>
<td>*OPC?</td>
<td>*OPC? on page 103</td>
</tr>
<tr>
<td>*OPT?</td>
<td>*OPT? on page 104</td>
</tr>
<tr>
<td>*RCL</td>
<td>*RCL on page 104</td>
</tr>
<tr>
<td>*RST</td>
<td>*RST on page 104</td>
</tr>
<tr>
<td>*SAV</td>
<td>*SAV on page 104</td>
</tr>
<tr>
<td>*SRE[?]</td>
<td>*SRE[?] on page 105</td>
</tr>
<tr>
<td>*STB?</td>
<td>*STB? on page 105</td>
</tr>
<tr>
<td>*TST?</td>
<td>*TST? on page 105</td>
</tr>
<tr>
<td>*WAI</td>
<td>*WAI on page 106</td>
</tr>
</tbody>
</table>
**CLS**

Syntax: *CLS*

Description: This command clears all status register structures in a device. These registers include:
- OPERation Status Register structure
- QUESTionable Status Register structure
The corresponding enable registers are unaffected.

**CAL?**

Syntax: *CAL?

Description: This command returns calibration data.

**LRN?**

Syntax: *LRN?

Description: This command gets the device setup query. It returns the instrument settings by binary block data.

**ESE[?]**

Syntax: *ESE <NRf>

Description: The Standard Event Status Enable Command (*ESE) sets the Standard Event Enable Register. This register acts like a mask, so that the next time a selected bit goes high, the ESB bit in the status byte is set.
The query (*ESE?) returns the contents of the Standard Event Enable Register.

**ESR?**

Syntax: *ESR?

Description: This query returns the Standard Event Status Register content. The register is cleared after it is read.
*IDN*

**Syntax**
*IDN?*

**Description**
The IDEnntification query (*IDN?) response semantics are organized into four fields, separated by commas. The field definitions are as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>Keysight Technologies</td>
</tr>
<tr>
<td>Model</td>
<td>M80xx</td>
</tr>
<tr>
<td>Serial number</td>
<td>MYxxxxxxxx</td>
</tr>
<tr>
<td>Firmware level</td>
<td>X.x.x.xxx</td>
</tr>
</tbody>
</table>

*OPC*

**Syntax**
*OPC*

**Description**
The *OPC* command sets the Operation Complete (OPC) bit of the Standard Event Status Register (ESR) when the No Operation Pending flag is TRUE (No Operation Pending flag is attached to each overlapped command). Until that time, the controller may continue to parse and execute previous commands. It is good technique, then, to periodically poll the OPC bit to determine if the overlapped command has completed.

*OPC?*

**Syntax**
*OPC?*

**Description**
The *OPC?* query returns the ASCII character "1" in the Output Queue when the No Operation Pending flag is TRUE. At the same time, it also sets the Message Available (MAV) bit in the Status Byte Register. The *OPC?* will not allow further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.
**OPT?**

Syntax  
`*OPT?`

Description  
The `*OPT?` query returns the installed options for each installed module. If no options are installed, only the model number(s) of the installed module(s) are returned.

Example  
`*OPT?`

```
(M1-M8041A c08,c16)(M2-M8061A,004,008)
```

**RCL**

Syntax  
`*RCL`

Description  
The `*RCL` command recalls a predefined read only instrument state setting. This command recalls the same device-specific functions effected by the `*RST` and `*SAV` commands.

The range is `*RCL 1` to `*RCL 10`.

**RST**

Syntax  
`*RST`

Description  
The Reset Command (`*RST`) sets the device-specific functions to a known state that is independent of the past-use history of the device.

**SAV**

Syntax  
`*SAV`

Description  
The `*SAV` command saves a predefined read only instrument state setting. This command saves the same device-specific functions effected by the `*RST` and `*RCL` commands.

The range is `*SAV 1` to `*SAV 10`. 
**SRE[?]**

Syntax: *SRE[?] <NRf>

Description: The Service Request Enable Command (*SRE) sets the Service Request Enable Register. This acts as a mask on the Status Byte, defining when the instrument can issue a service request. For a service request to be issued, the summary bit in the Status Byte must match the bit in the Service Request Enable Register. More than one bit may be set by the *SRE command.

The query returns the current contents of the Service Request Enable Register.

**STB?**

Syntax: *STB?

Description: The Read Status Byte Query (*STB?) allows the programmer to read the status byte and Master Summary Status bit. When the status byte is read using the *STB command, bit 6 of the status byte is referred to as the Master Summary (MSS) bit. With this query, the status byte is not cleared when the value is read. It always reflects the current status of all the instrument's status registers.

**TST?**

Syntax: *TST?

Description: The self-test query starts all internal self-tests and places a response into the output queue indicating whether or not the device completed the self-tests without any detected errors. It returns a 0 for success; a 1 if a failure was detected.

Upon successful completion of *TST?, the device settings are restored to their values prior to the *TST?
*WAI

Syntax  *WAI

Description  The *WAI command allows no further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.

The *WAI command can be used for overlapped commands. It stops the program execution until any pending overlapped commands have finished. Specifically, it waits until the No Operation Pending flag is TRUE, or receipt of a dcas message, or a power on.
SCPI Standard Commands

The following SCPI standard commands are implemented:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SYSTem:ERRor[:NEXT]?</td>
<td>Queries and at the same time deletes the oldest entry in the error queue.</td>
</tr>
<tr>
<td>:SYSTem:HELP:HEADers?</td>
<td>This query returns all SCPI commands, queries and IEEE 488.2 common commands and queries currently implemented in the instrument.</td>
</tr>
<tr>
<td>:SYSTem:VERSion?</td>
<td>This query returns a numeric value corresponding to the SCPI version number for which the instrument complies. The format is YYYY.V where Ys correspond to the year and the V corresponds to an approved version number for that year.</td>
</tr>
</tbody>
</table>
Miscellaneous Commands

The following miscellaneous commands are implemented:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
</table>

:TEST:RESults?

Syntax   :TEST:RESults?

Description This command returns the results of self test.
Command Syntax to Find Min/Max Values

The following example illustrates the syntax to find min/max values for any system parameter:

In the below example, "Voltage Amplitude" is the parameter for which min/max values are to be found.

:SOURce:VOLTage:AMPLitude?

Syntax for Min/Max Value
- :SOURce:VOLTage:AMPLitude? 'identifier', <MIN/MAX>

Example for Min Value

Example for Max Value
## Location and Module Mapping - Reference

The following table lists locations and their respective modules.

<table>
<thead>
<tr>
<th>Location</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Out</td>
<td>M8041A, M8051A, M8061A, M8062A, M8195A and M8045A</td>
</tr>
<tr>
<td>Data In</td>
<td>M8041A, M8051A, M8061A, M8062A and M8046A</td>
</tr>
<tr>
<td>System</td>
<td>M8041A</td>
</tr>
<tr>
<td>Clk Gen</td>
<td>M8041A, M8061A, M8195A and M8045A</td>
</tr>
<tr>
<td>Ref Clk Out</td>
<td>M8041A and M8045A</td>
</tr>
<tr>
<td>Clk Out</td>
<td>M8041A and M8045A</td>
</tr>
<tr>
<td>Cln Clk Out</td>
<td>M8062A</td>
</tr>
<tr>
<td>Trig Out</td>
<td>M8041A and M8045A</td>
</tr>
<tr>
<td>Sys Out A</td>
<td>M8041A and M8045A</td>
</tr>
<tr>
<td>Sys Out B</td>
<td>M8041A and M8045A</td>
</tr>
<tr>
<td>Ctrl Out A</td>
<td>M8041A, M8051A, M8045A and M8046A</td>
</tr>
<tr>
<td>Sys In A</td>
<td>M8041A and M8045A</td>
</tr>
<tr>
<td>Sys In B</td>
<td>M8041A and M8045A</td>
</tr>
<tr>
<td>Ctrl In A</td>
<td>M8041A, M8051A, M8045A and M8046A</td>
</tr>
<tr>
<td>Ctrl In B</td>
<td>M8041A and M8051A</td>
</tr>
<tr>
<td>Elect Idle In</td>
<td>M8061A and M8062A</td>
</tr>
<tr>
<td>Clk In</td>
<td>M8046A</td>
</tr>
</tbody>
</table>

**NOTE**
The “System” location is only available for few SCPIs. It is not visible on the M8070B user interface.
STATus Subsystem

This subsystem has the following SCPI structure:

```
:STATus
  :QUESTionable
  ...
  :OPERation
  ...
  :INSTRument
  ...
```

This subsystem has the following commands and subnodes:

Table 26

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under Subnodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>:QUESTionable</td>
<td>:STATus:QUESTionable Subnode on page 112</td>
</tr>
<tr>
<td>:OPERation</td>
<td>:STATus:OPERation Subnode on page 121</td>
</tr>
<tr>
<td>INSTRument</td>
<td>:STATus:INSTRument Subnode on page 125</td>
</tr>
</tbody>
</table>
:STATus:QUESTionable Subnode

This subnode has the following SCPI structure:

```
:STATus
  :QUESTionable
    :CONDition?
    :ENABLE[?]
    [:EVENT]?
    :NTRansition[?]
    :PTRansition[?]
    :INSTrument[?]
    :CONDition?
    :ENABLE[?]
    [:EVENT]?
    :NTRansition[?]
    :PTRansition[?]
    :PROtection
    :CONDition?
    :ENABLE[?]
    [:EVENT]?
    :NTRansition[?]
    :PTRansition[?]
    :SYMBOL
    :CONDition?
    :ENABLE[?]
    [:EVENT]?
    :NTRansition[?]
    :PTRansition[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:CONDition?</td>
<td>:STATus:QUESTionable:CONDition? on page 113</td>
</tr>
<tr>
<td>:ENABLE[?]</td>
<td>:STATus:QUESTionable:ENABLE[?] on page 114</td>
</tr>
<tr>
<td>[:EVENT]?</td>
<td>:STATus:QUESTionable[:EVENT]? on page 114</td>
</tr>
<tr>
<td>Name</td>
<td>Description under</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>:INSTrument?</td>
<td>:STATus:QUEStionable:INSTrument on page 115</td>
</tr>
<tr>
<td>:INSTrument[:EVENT]?</td>
<td>:STATus:QUEStionable:INSTrument[:EVENT]? on page 116</td>
</tr>
<tr>
<td>:PROTection</td>
<td>:STATus:QUEStionable:PROTection on page 117</td>
</tr>
<tr>
<td>:PROTection[:EVENT]?</td>
<td>:STATus:QUEStionable:PROTection[:EVENT]? on page 118</td>
</tr>
<tr>
<td>:SYMBol</td>
<td>:STATus:QUEStionable:SYMBol on page 119</td>
</tr>
<tr>
<td>:SYMBol[:EVENT]?</td>
<td>:STATus:QUEStionable:SYMBol[:EVENT]? on page 119</td>
</tr>
</tbody>
</table>

**:STATus:QUEStionable:CONDition?**

**Syntax**: :STATus:QUEStionable:CONDition?

**Description**: This command query returns the contents of the condition register of the Questionable Status Register structure.
:STATus:QUESTionable:ENABle[?]

Syntax   :STATus:QUESTionable:ENABle <NRf>
:STATus:QUESTionable:ENABle?

Input   <NRf>: Set enable mask.

Parameters

Description   This command sets the enable mask in the Questionable Status Register structure, which allows true conditions in the event register to be reported in the summary bit. The query returns the weighted value of the bits that are set in the enable register.

:STATus:QUESTionable[:EVENt]?

Syntax   :STATus:QUESTionable:EVENt?

Description   This command query returns the contents of the Questionable Status event register.

:STATus:QUESTionable:NTRansition[?]

Syntax   :STATus:QUESTionable:NTRansition <NRf>
:STATus:QUESTionable:NTRansition?

Input   <NRf>: Set transition filter state.

Parameters

Description   This command sets the transition filter state in the Questionable Status Register structure. The query returns the weighted value of the bits that are set to pass negative transitions in the transition filter.

:STATus:QUESTionable:PTRansition[?]

Syntax   :STATus:QUESTionable:PTRansition <NRf>
:STATus:QUESTionable:PTRansition?

Input   <NRf>: Set transition filter state.

Parameters
Description
This command sets the transition filter state in the Questionable Status Register structure. This is the default setting of the instrument. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter.

:STATus:QUEStionable:INSTrument

The summary bit of this register set is reflected in bit 9 of the Questionable status register.

The :STATus:QUEStionable:INSTrument contains conditions which reflect states of the instrument’s normal operation. These bits are summary bits. For example, if a Data Loss is detected by one or more error detectors, the Data Loss condition will be set.

The definition of each of these bits (condition register) is as follows:

**Bit 0 – Error**
One or more error detectors detect an error.

**Bit 1 – Data Loss**
This bit is set when the data source is turned off, not connected, or the cables or device is faulty. This bit can also be set when the 0/1 threshold is not within the eye limits of the incoming data signal. In this last case, use Auto Align or select Avg 0/1 Threshold.

**Bit 2 – Sync Loss**
This bit is set when the error detector pattern does not match the incoming data pattern or the BER of your device is higher than the sync threshold.

**Bit 3 – Clock Loss**
This bit is set when the pattern generator receives no external clock signal or the error detector receives no clock input signal.

**Bit 4 – CDR Unlocked**
One or more CDRs of the instrument are unlocked.
:STATus:QUEStionable:INSTrument:CONDition?
Syntax :STATus:QUEStionable:INSTrument:CONDition?
Description This command returns the contents of the condition register of the Questionable Status Register (bit 9), which reflects the state of the instrument.

:STATus:QUEStionable:INSTrument:ENABle[?] 
Syntax :STATus:QUEStionable:INSTrument:ENABle <NRf> 
:STATus:QUEStionable:INSTrument:ENABle?
Description This command sets the bits in the event enable register that can generate a summary bit used for the instrument state. The query returns the weighted value of the bits that are set in the event enable register.

:STATus:QUEStionable:INSTrument[:EVENt]? 
Syntax :STATus:QUEStionable:INSTrument[:EVENt]?
Description This command queries the contents of the Questionable Status event register.

:STATus:QUEStionable:INSTrument:NTRansition[?] 
Syntax :STATus:QUEStionable:INSTrument:NTRansition[?] <NRf> 
:STATus:QUEStionable:INSTrument:NTRansition?
Input Parameters <NRf>: Set transition filter state.
Description This command sets the transition filter state in the Questionable Status Register structure. The query returns the weighted value of the bits that are set to pass negative transitions in the transition filter.
:STATus:QUEStionable:INSTrument:PTRansition[?]

Syntax: :STATus:QUEStionable:INSTrument:PTRansition[?] <NRf>
:STATus:QUEStionable:INSTrument:PTRansition?

Input Parameters:
- <NRf>: Set transition filter state.

Description: This command sets the transition filter state in the Questionable Status Register structure. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter.

:STATus:QUEStionable:PROTection

The summary bit of this register set is reflected in bit 11 of the Questionable Status Register.

The :STATus:QUEStionable:PROTection contains conditions which reflect states of the instrument's normal operation. These are summary bits. For example, if one or more outputs of the instrument detect an overload condition, the OverloadDetection bit will be set.

The definition of each of these bits (condition register) is as follows:

**Bit 0 - OverloadDetection**
Indicates that an overload condition has been detected.

:STATus:QUEStionable:PROTection:CONDition?

Syntax: :STATus:QUEStionable:PROTection:CONDition?

Description: This command returns the contents of the condition register of the Questionable Status Register (bit 11) to detect an overload condition.
:STATus:QUESTionable:PROTection:ENABle?

Syntax  
:STATus:QUESTionable:PROTection:ENABle<NRf>
:STATus:QUESTionable:PROTection:ENABle?

Description  
This command sets the bits in the event enable register that can generate a summary bit used for monitoring a symbol alignment loss. The query returns the weighted value of the bits that are set in the event enable register.

:STATus:QUESTionable:PROTection[:EVENt]?

Syntax  
:STATus:QUESTionable:PROTection[:EVENt]?

Description  
This command queries the contents of the Questionable Status event register.

:STATus:QUESTionable:PROTection:NTRansition[

Syntax  
:STATus:QUESTionable:PROTection:NTRansition<NRf>
:STATus:QUESTionable:PROTection:NTRansition?

Input Parameters  
<NRf>: Set transition filter state.

Description  
This command sets the transition filter state in the Questionable Status Register structure. The query returns the weighted value of the bits that are set to pass negative transitions in the transition filter.

:STATus:QUESTionable:PROTection:PTRansition[

Syntax  
:STATus:QUESTionable:PROTection:PTRansition<NRf>
:STATus:QUESTionable:PROTection:PTRansition?

Input Parameters  
<NRf>: Set transition filter state.

Description  
This command sets the transition filter state in the Questionable Status Register structure. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter.
The summary bit of this register set is reflected in bit 12 of the Questionable Status Register.

The :STATus:QUEStionable:SYMBol contains conditions which reflect states of the instrument's normal operation. These are summary bits. For example, if an error detector(s) of the instrument detects a symbol alignment loss, the SymbolAlignmentLoss bit will be set.

The definition of each of these bits (condition register) is as follows:

**Bit 0 - SymbolAlignmentLoss**
Indicates whether the error detector(s) has experienced a symbol alignment loss.

---

### :STATus:QUEStionable:SYMBol:CONDition?

**Syntax**
:STATus:QUEStionable:SYMBol:CONDition?

**Description**
This query returns the contents of the condition register of the Questionable Status Register (bit 12) to detect a symbol alignment loss.

### :STATus:QUEStionable:SYMBol:ENABle[?]

**Syntax**
:STATus:QUEStionable:SYMBol:ENABle <NRf>
:STATus:QUEStionable:SYMBol:ENABle?

**Description**
This command sets the bits in the event enable register that can generate a summary bit used for monitoring a symbol alignment loss. The query returns the weighted value of the bits that are set in the event enable register.

### :STATus:QUEStionable:SYMBol[:EVENt]?

**Syntax**
:STATus:QUEStionable:SYMBol[:EVENt]?

**Description**
This command queries the contents of the Questionable Status event register.
:STATus:QUESTIONable:SYMBol:NTRansition[?]
Syntax   :STATus:QUESTIONable:SYMBol:NTRansition <NRf>
          :STATus:QUESTIONable:SYMBol:NTRansition?
Input    <NRf>: Set transition filter state.
Parameters
Description This command sets the transition filter state in the Questionable Status
          Register structure. The query returns the weighted value of the bits that
          are set to pass negative transitions in the transition filter.

:STATus:QUESTIONable:SYMBol:PTRansition[?]
Syntax   :STATus:QUESTIONable:SYMBol:PTRansition <NRf>
          :STATus:QUESTIONable:SYMBol:PTRansition?
Input    <NRf>: Set transition filter state.
Parameters
Description This command sets the transition filter state in the Questionable Status
          Register structure. The query returns the weighted value of the bits that
          are set to pass positive transitions in the transition filter.
:STATus:OPERation Subnode

This subnode has the following SCPI structure:

```
:STATus
   OPERation
     :CONDition?
     :ENABLE[?]?
     [:EVEN]??
     :NTRansition[?]
     :PTRansition[?]
   RUN
     :CONDition?
     :ENABLE[?]
     [:EVEN]??
     :NTRansition[?]
     :PTRansition[?]
```

Table 28

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ENABLE[?]</td>
<td>:STATus:OPERation:ENABLE[?] on page 122</td>
</tr>
<tr>
<td>[:EVEN]?</td>
<td>:STATus:OPERation[:EVENT]? on page 122</td>
</tr>
<tr>
<td>:NTRansition[?]</td>
<td>:STATus:OPERation:NTRansition[?] on page 122</td>
</tr>
<tr>
<td>:PTRansition[?]</td>
<td>:STATus:OPERation:PTRansition[?] on page 123</td>
</tr>
<tr>
<td>:RUN</td>
<td>:STATus:OPERation:RUN on page 123</td>
</tr>
<tr>
<td>:RUN[:EVENT]?</td>
<td>:STATus:OPERation:RUN[:EVENT]? on page 124</td>
</tr>
</tbody>
</table>
:STATus:OPERation:CONDition?
Syntax :STATus:OPERation:CONDition?
Description This query returns the contents of the condition register of the Operation Status Register structure.

:STATus:OPERation:ENABLE[?]
Syntax :STATus:OPERation:ENABLE <NRf>
:STATus:OPERation:ENABLE?
Input Parameters <NRf>: Set enable mask.
Description This command sets the enable mask in the Operation Status Register structure, which allows true conditions in the event register to be reported in the summary bit. The query returns the weighted value of the bits that are set in the enable register.

:STATus:OPERation[:EVENT]?
Syntax :STATus:OPERation[:EVENT]?
Description This query returns the content of Operation Status Event Register.

:STATus:OPERation:NTRansition[?]
Syntax :STATus:OPERation:NTRansition <NRf>
:STATus:OPERation:NTRansition?
Input Parameters <NRf>: Set transition filter state.
Description This command sets the transition filter state in the Operation Status Register structure. The query returns the weighted value of the bits that are set to pass negative transitions in the transition filter.
:STATus:OPERation:PTRansition

Syntax
:STATus:OPERation:PTRansition <NRf>
:STATus:OPERation:PTRansition?

Input
<NRf>: Set transition filter state.

Parameters
Description
This command sets the transition filter state in the Operation Status Register structure. This is the default setting of the instrument. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter.

:STATus:OPERation:RUN

The summary bit of this register set is reflected in bit 8 of the Operation Status Register.

The :STATus:OPERation:RUN contains conditions which reflect states of the instrument's normal operation. These are summary bits. For example, if the pattern generators, error detectors and clock generator are functioning normally, the Running bit will be set.

The definition of each of these bits (condition register) is as follows:

**Bit 0 - Running**

Indicates that the instrument is in normal operation mode; generators are working, all error detectors are running and the clock generator is functioning properly.

:STATus:OPERation:RUN:CONDition?

Syntax
:STATus:OPERation:RUN:CONDition?

Description
This command returns the contents of the condition register of the Operation Status Register (bit 8) to detect if all components are operating normally.
:STATus:OPERation:RUN:ENABle[?]  
Syntax :STATus:OPERation:RUN:ENABle <NRf>  
:STATus:OPERation:RUN:ENABle?  
Description This command sets the bits in the event enable register that can generate a summary bit used to determine if the hardware is functioning properly. The query returns the weighted value of the bits that are set in the event enable register.

:STATus:OPERation:RUN[:EVENt]?  
Syntax :STATus:OPERation:RUN[:EVENt]?  
Description This query returns the contents of the Operation Status event register.

:STATus:OPERation:RUN:NTRansition[?]  
Syntax :STATus:OPERation:RUN:NTRansition <NRf>  
:STATus:OPERation:RUN:NTRansition?  
Input Parameters <NRf>: Set transition filter state.  
Description This command sets the transition filter state in the Operation Status Register structure. The query returns the weighted value of the bits that are set to pass negative transitions in the transition filter.

:STATus:OPERation:RUN:PTRansition[?]  
Syntax :STATus:OPERation:RUN:PTRansition <NRf>  
:STATus:OPERation:RUN:PTRansition?  
Input Parameters <NRf>: Set transition filter state.  
Description This command sets the transition filter state in the Operation Status Register structure. The query returns the weighted value of the bits that are set to pass positive transitions in the transition filter.
:STATus:INStrument Subnode

The :STATus:INStrument commands return the 'current' value/condition of several predefined device states. These commands are a set of SCPI queries which return true (1) or false (0).

This subnode has the following SCPI structure:

```
:STATus
  :INStrument
    :CLOS?e?
    :DLOSs?
    :SLOSs?
    :SALoss?
    :ERRor?
    :OVERload?
    :RCINput?
    :RUN?
    :WAIT?
    :CDR
    :ULOCKed?
```

This subnode has the following commands:

**Table 29**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ERRor?</td>
<td>:STATus:INStrument:ERRor? on page 127</td>
</tr>
</tbody>
</table>
### :STATUs:INSTRument:CLOSs?

**Syntax**

:STATUs:INSTRument:CLOSs? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DataIn1' or 'M*.DataIn2' or 'M*.DataOut1' or 'M*.DataOut2' or 'M1.System'

**Return Range**

0|1

**Description**

This query indicates if the clock has been lost. A true (1) indicates a clock loss.

**Example**

:STAT:INST:CLOS? 'M1.DataIn1'

### :STATUs:INSTRument:DLOSs?

**Syntax**

:STATUs:INSTRument:DLOSs? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DataIn1' or 'M*.DataIn2' or 'M*.DataOut1' or 'M*.DataOut2' or 'M1.System'

**Return Range**

0|1

**Description**

This query indicates if the data has been lost. A true (1) indicates data loss.

**Example**

:STAT:INST:DLOS? 'M1.DataIn1'
:STATus:INStrument:SLOSs?

Syntax   :STATus:INStrument:SLOSs? 'identifier'
Input    'identifier': 'M*.DataIn1' or 'M*.DataIn2' or 'M1.System'
Parameters  Return Range
            0|1
Description This query indicates if the sync has been lost. A true (1) indicates a sync loss.
Example   :STAT:INST:SLOS? 'M1.DataIn1'

:STATus:INStrument:SALoss?

Syntax   :STATus:INStrument:SALoss? 'identifier'
Input    'identifier': 'M*.DataIn1' or 'M*.DataIn2' or 'M1.System'
Parameters  Return Range
            0|1
Description This query indicates if the symbol alignment has been lost. A true (1) indicates a symbol alignment loss.
Example   :STAT:INST:SAL? 'M1.DataIn1'

:STATus:INStrument:ERRor?

Syntax   :STATus:INStrument:ERRor? 'identifier'
Input    'identifier': 'M*.DataIn1' or 'M*.DataIn2' or 'M1.System'
Parameters  Return Range
            0|1
Description This query indicates if an error condition has occurred. A true (1) indicates an error condition.
Example   :STAT:INST:ERR? 'M1.DataIn1'
:STAT:INSTR:OVERload?
Syntax :STAT:INSTR:OVERload? 'identifier'
Input 'identifier': 'M*.DataIn1' or 'M*.DataIn2' or 'M*.DataOut1' or 'M*.DataOut2'
Parameters or 'M1.System'
Return Range 0|1
Description This query indicates an overload condition exists at an input or output. A true (1) indicates an overload condition.
Example :STAT:INST:OVER? 'M1.DataIn1'

:STAT:INSTR:RCInput?
Syntax :STAT:INSTR:RCInput? 'identifier'
Input 'identifier': 'M1.ClkGen'
Parameters
Return Range 0|1
Description This query indicates the input state of the reference clock.

:STAT:INSTR:RUN?
Syntax :STAT:INSTR:RUN? 'identifier'
Input 'identifier': 'M*.DataIn1' or 'M*.DataIn2' or 'M1.System'
Parameters
Return Range 0|1
Description This query indicates if the addressed channel is in running state. A true (1) indicates the input is in running state.
Example :STAT:INST:RUN? 'M1.DataIn1'
:STATus:INSTRument:RUN:WAIT?

Syntax :STATus:INSTRument:RUN:WAIT? 'identifier'[timeout]

Input Parameters

Parameters

Description This query waits until the addressed channel is in running state. Using this query is recommended before starting any measurement that relies on the operation of the addressed channel.

The optional timeout argument is having a default value of 90 and the accepted value range is 2 ... 300. The timeout is specified in terms of seconds.

- The VISA timeout must be set to at least the timeout value that is used for :STAT:INST:RUN:WAIT?
- *OPC? does not wait until the individual channels are running, since this cannot be ensured for channels that are clocked by externally provided signals, or for error detector channels that operate on a recovered clock signal.

Example Change the system frequency and wait until the pattern generator channels are running again


:STATus:INSTRument:CDR:ULOCked?

Syntax :STATus:INSTRument:CDR:ULOCked? 'identifier'

Input Parameters

Return Range 0|1

Description This query indicates the lock state of the clock data recovery. A true (1) indicates an unlocked state.

Example :STAT:INST:CDR:ULOC? 'M1.DataIn1'
:STATus:INStrument:ALIGNment Subnode

This subsystem is reporting alignment specific status information, and is offers an alternative way of retrieving the same status information that is available via the :INPut:ALIGNment:STATus Subnode on page 277.

This subsystem has the following SCPI structure:

```
:STATus
    :INStrument
        :ALIGNment[:VALue]?
```

Table 30

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:STATus:INStrument:ALIGNment[:VALue]?</td>
<td>:STATus:INStrument:ALIGNment[:VALue]? on page 130</td>
</tr>
</tbody>
</table>

:STATus:INStrument:ALIGNment[:VALue]?

**Syntax**: :STATus:INStrument:ALIGNment[:VALue]? 'identifier',

**Input**: 'identifier': Any error detector location and group name. E.g. 'M1.DataIn1', 'M2.DataIn', 'INF1.DataIn1'

**Parameters**: 'M2.DataIn1', 'INF1.DataIn1'

**Description**: This command retrieves alignment status information for an error detector or error detector group.

This command can be used to retrieve information about the alignment procedure that is working on the addressed error detector channel, or a group of error detectors. This command is reporting the status independently of the executing alignment procedure. There is no differentiation in respect to different types of alignments.

This command returns the following values:

- **INPROGRESS** - At least one addressed error detector is executing an alignment procedure.
- **ABORTED** - The alignment procedure has been aborted by the user before all aligning error detector channels finished the procedure.
- **FAILED** - All addressed error detector channels have finished the alignment procedure, but at least one was unable to finish successfully.
- **SUCCESS** - The addressed error detector channels have successfully finished the alignment procedure. Error detector channels that are addressed by the command but return **UNKNOWN** state, do not contribute to the result.
- **UNKNOWN** - None of the addressed error detectors has executed an alignment procedure yet.

**NOTE**

This command cannot be used in the same compound SCPI statement that is starting the alignment procedure. If this command is sent to the instrument right after the command that is initiating the alignment procedure, then the returned value is undefined and can reflect the state that was valid before the alignment. This can be overcome by reading any value from the instrument (e.g. *ESE?), except for *OPC? as this will block until the alignment procedure is finished.

**Example**: 
:INP:INST:ALIGN? 'M1.DataIn1'
The TRIGger subsystem selects the trigger mode at the TRIG OUT port. This subsystem has the following SCPI structure:

```
:TRIGger
    [:SOURce][?]
    [:FREQuency]
    [:INTernal
        [:SOURce][?]
    ]
    [:REFerence
        [:FREQuency][?]
    ]
    [:DIRect
        [:FREQuency][?]
        [:DETect]
    ]
    [:CMULtiplier
        [:LBWidth][?]
        [:FREQuency][?]
        [:DETect
            [:MULTiplier][?]
        ]
        [:DIVider][?]
    ]
```

Table 31

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:SOURce]</td>
<td>:TRIGger[:SOURce][?] on page 133</td>
</tr>
<tr>
<td>[:SOURce]:FREQuency?</td>
<td>:TRIGger[:SOURce]:FREQuency? on page 134</td>
</tr>
<tr>
<td>[:INTernal[:SOURce][?]</td>
<td>:TRIGger:INTernal[:SOURce][?] on page 134</td>
</tr>
<tr>
<td>[:REFerence[:FREQuency][?]</td>
<td>:TRIGger:REFerence[:FREQuency][?] on page 135</td>
</tr>
<tr>
<td>[:DIRect[:FREQuency][?]</td>
<td>:TRIGger:DIRect[:FREQuency][?] on page 135</td>
</tr>
<tr>
<td>[:DIRect[:FREQuency]:DETect</td>
<td>:TRIGger:DIRect[:FREQuency][?] on page 135</td>
</tr>
<tr>
<td>[:CMULtiplier:LBWidth[?]</td>
<td>:TRIGger:CMULtiplier:LBWidth[?] on page 136</td>
</tr>
</tbody>
</table>
**:TRIGger[:SOURce][?]**

**Syntax**  
:TRIGger[:SOURce] 'identifier', INTernal|REference|DIRect|CMULtiplier  
:TRIGger[:SOURce]? 'identifier'

**Input Parameters**  
'identifier': 'M1.ClkGen', 'INF*.Common'

**Return Range**  
INT|REF|DIR|CMUL

**Description**  
Selects the different trigger modes. The query returns the current trigger source mode.

The trigger source modes include the following:

- INTernal 100 MHz reference clock or AXIe 100 MHz (M9505A AXIe Chassis)
- REference 10 MHz / 100 MHz external reference clock
- DIRect 8.1 GHz to 16.207 GHz clock used as the system frequency directly in the M8041A
- CMULtiplier Multiplied/divided clock used to specify system frequency

This SCPI is applicable for M8041A, M8062A, M8045A, DSO634A, DSA634A, DSA96204A and DSOX96204A.

**Example**  
:TRIGger:SOURce 'M1.ClkGen', INT  
:TRIGger:SOURce? 'M1.ClkGen'
:TRIGger[:SOURce]:FREQuency?

Syntax: :TRIGger[:SOURce]:FREQuency? 'identifier'

Parameters:
- 'identifier': 'M1.TrigOut'

Description: This query returns the effective frequency of the trigger out clock signal. This parameter is only active if the subrate clock operating mode is selected. The mentioned frequency value depends upon the system frequency & divider value.

This SCPI is applicable for M8041A.

Example: :TRIG:FREQ? 'M1.TrigOut'

:TRIGger:INTernal[:SOURce][?]?

Syntax: :TRIGger:INTernal[:SOURce] 'identifier', <AXIFrame|INTernal>

Parameters:
- 'identifier': 'M1.ClkGen'

<AXIFrame>: Select 100 MHz reference of AXIFrame as reference clock.

<INTernal>: Select 100 MHz reference of the M8041A as reference clock.

Return Range: AXIF|INT

Description: In the INTernal mode different reference clock sources are available for the internal oscillator. Select between the 100 MHz reference clock of the M9505A AXIe Chassis or the 100 MHz reference clock of the M8041A module.

This SCPI is applicable for M8041A and M8045A.


### :TRIGger:REference[:FREQuency]?[]

**Syntax**

:TRIGger:REference[:FREQuency] 'identifier', <REF10|REF100>

:TRIGger:REference[:FREQuency]? 'identifier'

**Input**

'identifier': 'M1.ClkGen', 'INF*.Common'

**Parameters**

<R:REF10|REF100>: Set external reference clock to 10 MHz or 100 MHz.

**Return Range**

REF10|REF100

**Description**

In the REFerence mode two different expected external reference clocks can be specified: 10 MHz / 100 MHz.

This SCPI is applicable for M8041A, M8045A and DSO634A, DSA634A, DSAX96204A and DSOX96204A.

**Example**

:TRIGger:REference:FREQuency 'M1.ClkGen', REF100

### :TRIGger:DIRect[:FREQuency]?[]

**Syntax**

:TRIGger:DIRect[:FREQuency] 'identifier', <NRf>

:TRIGger:DIRect[:FREQuency]? 'identifier'

**Input**

'identifier': 'M1.ClkGen'

**Parameters**

<NRf>: Enter frequency value at REF CLK IN.

**Return Range**

8.1 GHz to 16.207 GHz

**Description**

In the DIRect mode the frequency value expected at the REF CLK IN port can be set manually. Acceptable units include MHz, GHz and exponents (for example, 10E9 is the same as 10 GHz).

This SCPI is applicable for M8041A and M8045A.

**Example**

:TRIG:DIR:FREQ 'M1.ClkGen',10e9
:TRIGger:DIRect[:FREQuency]:DETect
Syntax  :TRIGger:DIRect[:FREQuency]:DETect 'identifier'
Input   'identifier': 'M1.ClkGen'
Parameters
Description The externally provided clock on the REF CLK IN port is measured once and used as the new system frequency (data rate), as required.
This SCPI is applicable for M8041A and M8045A.
Example  :TRIG:DIR:DET 'M1.ClkGen'

:TRIGger:CMULtiplier:LBWidth[]
Syntax  :TRIGger:CMULtiplier:LBWidth 'identifier', <BW100|BW2|BW5>
:TRIGger:CMULtiplier:LBWidth? 'identifier'
Input   'identifier': 'M1.ClkGen'
Parameters  <BW100|BW2|BW5>: Specify loop bandwidth.
Return Range BW100|BW2|BW5
Description Different loop bandwidths can be selected. Modulated signals (for example SSC) at the REF CLK IN port can be provided depending on the loop bandwidth.
This SCPI is applicable for M8041A.
Example  :TRIG:CMUL:LBW 'M1.ClkGen', BW2

:TRIGger:CMULtiplier:FREQuency:MULTiplier[]
Syntax  :TRIGger:CMULtiplier:FREQuency:MULTiplier 'identifier', <NRf>
:TRIGger:CMULtiplier:FREQuency:MULTiplier? 'identifier'
Input   'identifier': 'M1.ClkGen'
Parameters  <NRf>: Specify the multiplier value.
Return Range 2 to 1620; the range is dependent on other system parameters. See Command Syntax to Find Min/Max Values on page 109.
This command specifies the external reference clock frequency multiplier.
This SCPI is applicable for M8041A and M8045A.

Example :TRIG:CMUL:FREQ:MULT 'M1.ClkGen',3

:TRIGger:CMULtiplier:FREQuency:DIVider[?]

Syntax :TRIGger:CMULtiplier:FREQuency:DIVider 'identifier', <NRf>
:TRIGger:CMULtiplier:FREQuency:DIVider? 'identifier'

Input
'identifier': 'M1.ClkGen'

Parameters
<NRf>: Specify divider value.

Return Range 1 to 63; Range is dependent on other system parameters.

This command specifies the external reference clock frequency divider.
This SCPI is applicable for M8041A and M8045A.

Example :TRIG:CMUL:FREQ:DIV 'M1.ClkGen',1

:TRIGger:CMULtiplier:FREQuency[?]

Syntax :TRIGger:CMULtiplier:FREQuency 'identifier', <NRf>
:TRIGger:CMULtiplier:FREQuency? 'identifier'

Input
'identifier': 'M1.ClkGen'

Parameters
<NRf>: Specify reference frequency value.

Return Range 10 MHz to 202.5875 MHz; the range is dependent on other system parameters. See Command Syntax to Find Min/Max Values on page 109.

This command is used to set the expected frequency value of the external provided clock at REF CLK IN manually. The sent value will be multiplied / divided for calculating the system frequency (data rate). Acceptable units include kHz, MHz and exponents (for example, 200E6 is the same as 200 MHz).
This SCPI is applicable for M8041A and M8045A.

Example :TRIG:CMUL:FREQ 'M1.ClkGen', 200e6
:TRIG:CMUL:FREQ? 'M1.ClkGen'
:TRIGger:CMULtiplier:FREQuency:DETect

Syntax  :TRIGger:CMULtiplier:FREQuency:DETect 'identifier'

Input   'identifier': 'M1.ClkGen'

Description  The external provided clock at the REF CLK IN port is measured once and will be multiplied/divided for calculating the system frequency (data rate).

              This SCPI is applicable for M8041A and M8045A.

Example  :TRIG:CMUL:FREQ:DET 'M1.ClkGen'
Source Subsystem

The SOURce subsystem controls output signals (for example, for setting frequency and levels).

This subsystem has the following SCPI structure:
This subsystem has the following commands and subnodes:

Table 32

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:FREQuency[?]</td>
<td>[:SOURce]:FREQuency[?] on page 141</td>
</tr>
<tr>
<td>:PERiod[?]</td>
<td>[:SOURce]:PERiod[?] on page 141</td>
</tr>
<tr>
<td>:DRATe[?]</td>
<td>[:SOURce]:DRATe[?] on page 142</td>
</tr>
<tr>
<td>:MEMory:CONFigure[:VALue][?]</td>
<td>[:SOURce]:MEMory:CONFigure[:VALue][?] on page 142</td>
</tr>
<tr>
<td>:MEMory:CONFigure:COUPling[?]</td>
<td>[:SOURce]:MEMory:CONFigure:COUPling[?] on page 143</td>
</tr>
<tr>
<td>Subnodes</td>
<td></td>
</tr>
<tr>
<td>:PULse</td>
<td>[:SOURce]:PULSe Subnode on page 144</td>
</tr>
<tr>
<td>:VOLTage</td>
<td>[:SOURce]:VOLTage Subnode on page 148</td>
</tr>
<tr>
<td>:JITter</td>
<td>[:SOURce]:JITTer Subnode on page 153</td>
</tr>
<tr>
<td>:JITter:HFRequency</td>
<td>[:SOURce]:JITTer:HFRequency Subnode on page 156</td>
</tr>
<tr>
<td>:JITter:LFRequency</td>
<td>[:SOURce]:JITTer:LFRequency Subnode on page 173</td>
</tr>
<tr>
<td>:JITter:SWEep</td>
<td>[:SOURce]:JITTer:SWEep Subnode on page 181</td>
</tr>
<tr>
<td>:SSCLocking</td>
<td>[:SOURce]:SSCLocking Subnode on page 187</td>
</tr>
<tr>
<td>:INTerference</td>
<td>[:SOURce]:INTerference Subnode on page 193</td>
</tr>
</tbody>
</table>
[:SOURce]:FREQuency[?]

Syntax   [:SOURce]:FREQuency 'identifier', <NRf>
[:SOURce]:FREQuency? 'identifier'

Input
'identifier': 'M1.ClkGen'

Parameters
<NRf>: Set system frequency.

Return Range
For M8041A: 256 MHz to 16.207 GHz
For M8195A: 256 MHz to 32.5 GHz
For M8045A: 2.025 GHz to 64.828 GHz

Description
Sets the frequency of the synthesizer in the M8020A/M8030A/M8040A.
Acceptable units include kHz, MHz, GHz and exponents (for example, 10E9 is the same as 10 GHz).

This SCPI is applicable for M8041A, M8195A, M8196A and M8045A.
The SCPI is read only for M8062A.

Example
:FREQ 'M1.ClkGen', 10e9

[:SOURce]:PERiod[?]

Syntax   [:SOURce]:PERiod 'identifier', <NRf>
[:SOURce]:PERiod? 'identifier'

Input
'identifier': 'M1.ClkGen'

Parameters
<NRf>: Set system period.

Return Range
For M8041A: 61.702 ps to 3.906249 ns
For M8195A: 30.77 ps to 3.90625 ns
For M8045A: 15.426 ps to 493.827

Description
Sets the M8020A/M8030A/M8040A system period. The period is the reciprocal value of the system frequency.
Acceptable units include ps, ns and exponents (for example, 61.702E-12 is the same as 61.702 ps).

This SCPI is applicable for M8041A, M8195A, M8196A and M8045A.

Example
:PER 'M1.ClkGen', 61.702ps
[:SOURce]:DRATe[?]

Syntax
[:SOURce]:DRATe 'identifier', <NRf>
[:SOURce]:DRATe? 'identifier'

Input
'identifier': 'M2.ClkGen'

Parameters
<NRF>: Set data rate.

Return Range 512 Mb/s to 32.414 Gb/s

Description This command/query sets/gets the data rate.
This SCPI is applicable for M8062A.

Example :DRAT 'M2.ClkGen', 10e9

[:SOURce]:MEMory:CONFigure[:VALue][?]

Syntax
[:SOURce]:MEMory:CONFigure 'identifier', <SINGle | DUAL | FOUR | RANDom | SINusoidal>
[:SOURce]:MEMory:CONFigure? 'identifier'

Input
'identifier': 'M1'

Parameters
SINGle | DUAL | FOUR | RANDom | SINusoidal

Case 1 - The following description is applicable when the four channel license has been purchased:

• SINGle: DataOut1 will be sourced from extended memory and the other channels will be sourced from module internal memory. The data rate range in this mode is 256 Mb/s ... 32.5 Gb/s.
• DUAL: DataOut1 and DataOut2 will be sourced from extended memory and the other channels will be sourced from module internal memory. The data rate range in this mode is 256 Mb/s ... 16.250 Gb/s.
• FOUR: All four channels will be sourced from extended memory. The data rate range in this mode is 256 Mb/s ... 8.125 Gb/s.
• RANDom: All the available channels are used to generate random interference.
• SINusoidal: All the available channels are used to generate random interference.
[:SOURce]:MEMory:CONFigure:COUPling[?]

Syntax [:SOURce]:MEMory:CONFigure:COUPling 'Identifier', NONe|ONETwo|ONEFour|THReeandfour

[:SOURce]:MEMory:CONFigure:COUPling? 'Identifier'

Input 'identifier': 'M*.DataOut1' or 'M*.DataOut2' or 'M*.DataOut3' or 'M*.DataOut4'

Parameters NONe | ONETwo | ONEFour | THReeandfour

Description This command sets the Channel Coupling for Channel One & Two and Channel Three & Four. In case of two channel license module, Channel One and Four will be coupled.

This query returns the current setting.

This SCPI is applicable for M8195A and M8196A.

:SOUR:MEM:CONF:COUP? 'M1'
ONET
This subnode has the following SCPI structure:

```
[:SOURce]
PULSe
  FILTer
    TYPE[?]
    ROFFfactor[?]
    DELay[?]
  TRANSition
    FIXed[?]
    LEADing[?]
```

This subnode has the following commands:

Table 33

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:PULSe:FILTER:TYPE[?]</td>
<td>[:SOURce]:PULSe:FILTER:TYPE[?] on page 144</td>
</tr>
<tr>
<td>:PULSe:ROFFfactor[?]</td>
<td>[:SOURce]:PULSe:ROFFfactor[?] on page 145</td>
</tr>
<tr>
<td>:PULSe:DELay[?]</td>
<td>[:SOURce]:PULSe:DELay[?] on page 145</td>
</tr>
<tr>
<td>:PULSe:TRANSition:FIXed[?]</td>
<td>[:SOURce]:PULSe:TRANSition:FIXed[?] on page 146</td>
</tr>
<tr>
<td>:PULSe:TRANSition:LEADing[?]</td>
<td>[:SOURce]:PULSe:TRANSition:LEADing[?] on page 146</td>
</tr>
</tbody>
</table>

`:SOURce]:PULSe:FILTER:TYPE[?]

Syntax [:SOURce]:PULSe:FILTER:TYPE 'identifier', <GAUS | RCOS>

Input Parameters

- <GAUS | RCOS>

Return Range  

GAUS | RCOS
Description

This command specifies filter type used on the waveform. Depending upon the specific waveform used, the waveform gets properties like transition time or bandwidth limits.

This query returns the present setting.

This SCPI is applicable for M8196A.

Example

[:SOURce]:PULSe:FILT:TYPE 'M1.DataOut2', GAUS

[:SOURce]:PULSe:ROFFfactor[?]

Syntax

[:SOURce]:PULSe:ROFFfactor 'identifier', <NRf>

[:SOURce]:PULSe:ROFFfactor? 'identifier'

Input Parameters

'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', or 'M*.DataOut4'

<NRF>: Sets the roll-off-factor value. The value should be between 0 to 1.

Description

This command sets the roll-off factor for the raised cosine pulse shaping filter. It is a measure for the excess bandwidth the waveform contains beyond Nyquist bandwidth.

This query returns the present setting.

This SCPI is applicable for M8196A.

Example

:PULS:ROFF 'M1.DataOut2', 0.50

[:SOURce]:PULSe:DElay[?]

Syntax

[:SOURce]:PULSe:DElay 'identifier', <NRf>

[:SOURce]:PULSe:DElay? 'identifier'

Input Parameters

'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', or 'M*.DataOut4'

<NRF>: Set the pulse delay time.

Return Range

For M8041A: 0.0 ps to 100 ns
For M8195A: -1 ns to 1 ns
For M8062A: 0 ps to 100 ns
For M8045A: 0 ps to 100 ns

Description

This command sets the time from the start of the period to the first edge of the pulse. Acceptable units include ps, ns and exponents (for example, 25E-9 is the same as 25 ns).
This SCPI is applicable for M8041A, M8051A, M8195A, M8062A and M8045A.

Example: :PULS:DEL 'M1.DataOut2', 25ns

[:SOURce]:PULSe:TRANsition:FIXed[?]

Syntax [:SOURce]:PULSe:TRANsition:FIXed 'identifier', <SMOoth | MODerate | STEep>

[:SOURce]:PULSe:TRANsition:FIXed? 'identifier'

Input 'identifier': 'M*.DATAOUT1', 'M*.DataOut2', or 'M*.DataOut'

Return Range SMOoth | MODerate | STEep

Description Use this command to set/read the transition time of the data output stream.

Following is the expected transition time (tr) for the below mentioned settings:
1. Smooth: 20 ps typical (20%-80%)
2. Moderate: 17 ps typical (20%-80%)
3. Steep: 12 ps typical (20%-80%)

This SCPI is applicable for M8041A, M8051A and M8062A.


[:SOURce]:PULSe:TRANsition[:LEADing][?]

Syntax [:SOURce]:PULSe:TRANsition[:LEADing] 'identifier', <NRf>

[:SOURce]:PULSe:TRANsition[:LEADing]? 'identifier'

Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3' and 'M*.DataOut4'

<NRf>: Specify transition time value.

Description Use this command to set/read the transition time of the data output stream. The fundamental units for transition time is seconds.

This SCPI is applicable for M8195A.
Example

:PULS:TRAN 'M1.DataOut2',15e-12
:PULS:TRAN? 'M1.DataOut2'

NOTE
It is not possible to define rise and fall times separately. In other words, the rise and fall time will be in the same range.
[:SOURCE]:VOLTage Subnode

This subnode has the following SCPI structure:

```
[:SOURCE]
  :VOLTage
    [:AMPLitude][?
    [:CFAMplitude][?
    [:OFFSET][?
    [:HIGH][?
    [:LOW][?
    [:RANGe
      [:SELECT][?
      [:AUTO][?
```

This subnode has the following commands:

Table 34

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:AMPLitude][?]</td>
<td>[:SOURCE]:VOLTage[:AMPLitude][?] on page 149</td>
</tr>
<tr>
<td>[:CFAMplitude][?]</td>
<td>[:SOURCE]:VOLTage:CFAMplitude[?] on page 149</td>
</tr>
<tr>
<td>[:OFFSET][?]</td>
<td>[:SOURCE]:VOLTage:OFFSET[?] on page 150</td>
</tr>
<tr>
<td>[:HIGH][?]</td>
<td>[:SOURCE]:VOLTage:HIGH[?] on page 150</td>
</tr>
<tr>
<td>[:LOW][?]</td>
<td>[:SOURCE]:VOLTage:LOW[?] on page 151</td>
</tr>
<tr>
<td>[:RANGe:SELECT][?]</td>
<td>[:SOURCE]:VOLTage:RANGe:SELECT[?] on page 151</td>
</tr>
<tr>
<td>[:RANGe:AUTO][?]</td>
<td>[:SOURCE]:VOLTage:RANGe:AUTO[?] on page 152</td>
</tr>
</tbody>
</table>
[:SOURce]:VOLTage[:AMPLitude]?  
Syntax  [:SOURce]:VOLTage[:AMPLitude] 'identifier', <NRf>  
       [:SOURce]:VOLTage[:AMPLitude]? 'identifier'  
Parameters  <NRf>: Set peak to peak voltage amplitude.  
Description  This command sets the peak to peak value of an output signal in Volts addressed by an identifier. Acceptable units include mV, V and exponents (for example, 500E-3 is the same as 500 mV).  
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8195A and M8045A.  
Example  :VOLT 'M1.DataOut1', 500mv  
          :VOLT? 'M1.DataOut1'  

[:SOURce]:VOLTage:CFAMplitude[?]  
Syntax  [:SOURce]:VOLTage:CFAMplitude 'Identifier', <NRf>  
       [:SOURce]:VOLTage:CFAMplitude? 'Identifier'  
Input  'identifier': 'M*.DataOut1' or 'M*.DataOut2' or 'M*.DataOut3' or 'M*.DataOut4'  
Parameters  'M*.DataOut4'  
Description  This command sets the amplitude correction factor for each channel. Amplitude correction factor parameter will be functional only in channel coupled mode. For details on how to set channel coupling mode, see [:SOURce]:MEMory:CONFigure:COUPling[?] on page 143.  
This SCPI is applicable for M8195A and M8196A.  
Example  :SOUR:VOLT:CFAM 'M1.DataOut1',1  
          :SOUR:VOLT:CFAM? 'M1.DataOut1'  
          1
[:SOURce]:VOLTage:OFFSet[?]  
**Syntax**  
[:SOURce]:VOLTage:OFFSet 'identifier', <NRf>  
[:SOURce]:VOLTage:OFFSet? 'identifier'  
**Input**  
**Parameters**  
<NRf>: Set offset voltage.  

**Return Range**  
<NRf>  

**Description**  
This command sets the offset value of an output signal in Volts addressed by an identifier. Acceptable units include mV, V and exponents (for example, 500E-3 is the same as 500 mV).  

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8195A and M8045A.  

**Example**  
:VOLT:OFFS 'M1.DataOut1', 500mv  
:VOLT:OFFS? 'M1.DataOut1'  

[:SOURce]:VOLTage:HIGH[?]  
**Syntax**  
[:SOURce]:VOLTage:HIGH 'identifier', <NRf>  
[:SOURce]:VOLTage:HIGH? 'identifier'  
**Input**  
**Parameters**  
<NRf>: Set upper voltage level.  

**Return Range**  
<NRf>  

**Description**  
This command sets the upper voltage level of an output signal in Volts addressed by an identifier. Acceptable units include mV, V and exponents (for example, 500E-3 is the same as 500 mV).  

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8195A and M8045A.  

**Example**  
:VOLT:HIGH 'M1.DataOut1', 500mv  
:VOLT:HIGH? 'M1.DataOut1'
[[:SOURce]:VOLTage:LOW[?]]

Syntax
[[:SOURce]:VOLTage:LOW 'identifier', <NRf>]
[[:SOURce]:VOLTage:LOW? 'identifier']

Input
'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', 'M*.DataOut4',
'M*.DataOut', or 'M*.CtrlOutA'

Parameters

<R1|R2|R3|R4|R5|R6|R7|R8>: Set lower voltage level.

Return Range

<NRf>

Description
This command sets the lower voltage level of an output signal in Volts
addressed by an identifier.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8195A
and M8045A.

Example
:VOLT:LOW 'M2.DataOut2', 100 mv
:VOLT:LOW? 'M2.DataOut2'

[[:SOURce]:VOLTage:RANGe[:SE lect]][?]

Syntax
[[:SOURce]:VOLTage:RANGe[:SE lect] 'identifier',
<R1|R2|R3|R4|R5|R6|R7|R8>]
[[:SOURce]:VOLTage:RANGe[:SE lect]? 'identifier']

Input
.identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters

<R1|R2|R3|R4|R5|R6|R7|R8>: Select amplitude ranges.

Return Range
R1|R2|R3|R4|R5|R6|R7|R8

Description
Amplitude ranges guarantee a glitch free change of the amplitude value
within a specified range. The upper and lower limits of these ranges are
specified in the data sheet. Ranges are only specified for the DataOut
channel.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

Example
:VOLT:RANG 'M2.DataOut2', R4
[[:SOURce]:VOLTage:RANGe:AUTO[?]]

Syntax  
[:SOURce]:VOLTage:RANGe:AUTO 'identifier', <ON|OFF|1|0>
[:SOURce]:VOLTage:RANGe:AUTO? 'identifier'

Parameters
'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

<ON|OFF|1|0>: Enable/disable auto tracking.

Return Range 1|0

Description Disable/enable the automatic tracking of the amplitude ranges.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

Example :VOLT:RANG:_AUTO 'M2.DataOut2', off
[:SOURce]:JITTer Subnode

This subnode has the following SCPI structure:

```
[:SOURce]
  :JITTer
    [:GLOBal]
    [:STATe][?]
    :CONFigure
    [:DELay][?]
    :HMODe[?]
    :HFRequency
    [...]
    :LFRequency
    [...]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:GLOBal][STATe][?]</td>
<td>[:SOURce]:JITTer[:GLOBal][STATe][?] on page 154</td>
</tr>
<tr>
<td>:CONFigure[:DELay][?]</td>
<td>[:SOURce]:JITTer:CONFigure[:DELay][?] on page 154</td>
</tr>
<tr>
<td>:CONFigure:HMODe[?]</td>
<td>[:SOURce]:JITTer:CONFigure:HMODe[?] on page 154</td>
</tr>
<tr>
<td>:HFRequency</td>
<td>[:SOURce]:JITTer:HFRequency Subnode on page 156</td>
</tr>
<tr>
<td>:LFRequency</td>
<td>[:SOURce]:JITTer:LFRequency Subnode on page 173</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer[:GLOBal][:STATe]?]

Syntax
[:SOURce]:JITTer[:GLOBal][:STATe] 'identifier', <ON|OFF|1|0>
[:SOURce]:JITTer[:GLOBal][:STATe]? 'identifier'

Input
'identifier': 'M1.System'

Parameters
<ON|OFF|1|0>: Enable/disable global state.

Return Range
1|0

Description
This command enables or disables global jitter.
It is a global SCPI; working for all modules.

Example
:JITT:STAT 'M1.System', off

[:SOURce]:JITTer:CONFigure[:DELay]?]

Syntax
[:SOURce]:JITTer:CONFigure[:DELay] 'identifier', <NRf>
[:SOURce]:JITTer:CONFigure[:DELay]? 'identifier'

Input
'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'

Parameters
<NRf>: Set jitter profile delay.

Return Range
-40 ns to 40 ns

Description
Sets the delay of the jitter profile of the corresponding output.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example

[:SOURce]:JITTer:CONFigure:HMODe?]

Syntax
[:SOURce]:JITTer:CONFigure:HMODe 'identifier', < ON | OFF | 1 | 0>
[:SOURce]:JITTer:CONFigure:HMODe? 'identifier'

Input
'identifier': 'M*.DataOut'

Parameters
< ON | OFF | 1 | 0>

Return Range
1 | 0
Description  
Sets the High Jitter Injection Mode of the corresponding output. 
This SCPI is applicable for M8062A.

Example  
JITT:CONF:HMOD 'M2.DataOut', OFF
This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:UNIT?]</td>
<td>[:SOURce]:JITTer:HFRequency:UNIT[?] on page 157</td>
</tr>
<tr>
<td>[:EXTernal[:STATe]?]</td>
<td>[:SOURce]:JITTer:HFRequency:EXTernal[:STATe] on page 157</td>
</tr>
<tr>
<td>::PMMode[?]</td>
<td>[:SOURce]:JITTer:HFRequency:PMMoed[?] on page 158</td>
</tr>
<tr>
<td>::PERiodic(*)[?]</td>
<td>[:SOURce]:JITTer:HFRequency:PERiodic(*)[:STATe][?] on page 160</td>
</tr>
<tr>
<td>:BUNCorrelate</td>
<td>[:SOURce]:JITTer:HFRequency:BUNCorrelate Subnode on page 162</td>
</tr>
<tr>
<td>:RANDom</td>
<td>[:SOURce]:JITTer:HFRequency:RANDom Subnode on page 166</td>
</tr>
<tr>
<td>:SPECTrally</td>
<td>[:SOURce]:JITTer:HFRequency:SPECTrally Subnode on page 170</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer:HFRequency:UNIT[?]

Syntax  [:SOURce]:JITTer:HFRequency:UNIT 'identifier', <UINTerval|TIME>
[:SOURce]:JITTer:HFRequency:UNIT? 'identifier'

Input  'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters  <UINTerval|TIME>: Specify jitter parameter units.

Return Range  UINT|TIME

Description  Specifies whether the jitter parameters are to be specified/returned in seconds (TIME) or unit intervals (UINTerval).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

This SCPI is also available on M8195A for DataOut1 to 4.

Example  :JITT:HFR:UNIT 'M1.DataOut1', TIME

[:SOURce]:JITTer:HFRequency:EXTernal[:STATe][?]

Syntax  [:SOURce]:JITTer:HFRequency:EXTernal[:STATe] 'identifier', <ON|OFF|1|0>
[:SOURce]:JITTer:HFRequency:EXTernal[:STATe]? 'identifier'

Input  'identifier': 'M1.ClkOut', 'M*.DataOut1' or 'M*.DataOut2'

Parameters  <ON|OFF|1|0>: Enable/disable external jitter input.

Return Range  1|0

Description  Enable/disable the external jitter input connector for Data Out1, Data Out2 and Clock Out. The name of the connectors are DATA MOD IN and CLK MOD IN.

This SCPI is applicable for M8041A, M8051A and M8045A.

Example  :JITT:HFR:EXT:STAT 'M1.DataOut2', on
[[:SOURce]:JITTer:HFRequency:PMMode]?]

Syntax  [:SOURce]:JITTer:HFRequency:PMMode 'identifier', <UI|TIME>
[[:SOURce]:JITTer:HFRequency:PMMode? 'identifier'

Input  'identifier': 'M*.DataOut1' or 'M*.DataOut2'
Parameters  <UI|TIME>: Specify unit interval or time.

Return Range  UI|TIME

Description  The internal delay line is optimized for either 'UI' (1UI; the maximum of the delay line is not used) or for 'TIME' rather than the whole range of the delay line, which can be used independent of the data rate. PMMode stands for Phase Modulation Mode.

This SCPI is applicable for M8041A, M8051A and M8045A.

Example  :JITT:HFR:PMM 'M1.DataOut2', UI
This subnode has the following SCPI structure:

```plaintext
[:SOURce]
  [:JITTer]
  [:HFRequency]
  [:PERiodic(*)]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:STATe]?</td>
<td>[:SOURce]:JITTer:HFRequency:PERiodic(*):STATe]? on page 160</td>
</tr>
<tr>
<td>:AMPLitude[?]</td>
<td>[:SOURce]:JITTer:HFRequency:PERiodic(*):AMPLitude[?] on page 160</td>
</tr>
<tr>
<td>:FREQuency[?]</td>
<td>[:SOURce]:JITTer:HFRequency:PERiodic(*):FREQuency[?] on page 161</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer:HFRequency:PERiodic(*)[:STATe]?

Syntax 
[:SOURce]:JITTer:HFRequency:PERiodic(*)[:STATe] 'identifier', <ON|OFF|1|0> 
[:SOURce]:JITTer:HFRequency:PERiodic(*)[:STATe]? 'identifier'

Input Parameters 
'identifier': 'M1.ClkOut', 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Return Range 1|0

Description 
This command is used to enable/disable periodic jitter high frequency output at the given location.

The "(*)" suffix specifies which of the two periodic sources (1 or 2) for the specified channel to use. If a suffix is not specified, the suffix 1 is assumed.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

This SCPI is also available on M8195A for DataOut1 to 4.

Example 
:JITT:HFR:PER 'M2.DataOut2', off

[:SOURce]:JITTer:HFRequency:PERiodic(*):AMPLitude?

Syntax 
[:SOURce]:JITTer:HFRequency:PERiodic(*):AMPLitude 'identifier', <NRf> 
[:SOURce]:JITTer:HFRequency:PERiodic(*):AMPLitude? 'identifier'

Input Parameters 
'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

<NRf>: Set periodic jitter amplitude.

Return Range 0 mUI to 1.102 UI; The range is dependent on other system parameters. See Command Syntax to Find Min/Max Values on page 109. Range for M8195A is 0 to 10 UI.

Description 
This command sets the amplitude of the periodic jitter in seconds or unit intervals for the specified output. The units are set using the [:SOURce]:JITTer:HFRequency:UNIT command.

The "(*)" suffix specifies which of the two periodic sources (1 or 2) for the specified channel to use. If a suffix is not specified, the suffix 1 is assumed.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.
This SCPI is also available on M8195A for DataOut1 to 4.

Example

```
:JITT:HFR:PER:AMPL 'M2.DataOut2', 100e-12
```

**URce]:JITTer:HFRequency:PERiodic(*):FREQueury[?]**

**Syntax**

```
[:SOURce]:JITTer:HFRequency:PERiodic(*):FREQueury 'identifier', <NRf>
[:SOURce]:JITTer:HFRequency:PERiodic(*):FREQueury? 'identifier'
```

**Input Parameters**

- `'identifier'`: 'M*.DataOut1' or 'M*.DataOut2', or 'M*.DataOut'
- `<NRf>`: Set periodic jitter frequency.

**Return Range**

1 kHz to 500 MHz

**Description**

This command sets the frequency of the periodic jitter for the specified output.

The "(*)" suffix specifies which of the two periodic sources (1 or 2) for the specified channel to use. If a suffix is not specified, the suffix 1 is assumed. Acceptable units include Hz, kHz, MHz and exponents (for example, 1E3 is the same as 1,000 Hz).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

This SCPI is also available on M8195A for DataOut1 to 4.

Example

```
```
This subnode has the following SCPI structure:

```
[:SOURce] [JITTter]
  [HFRequency]
    [BUNCorrelate]
      [:STATe][?]
      [:AMPLitude][?]
      [:DRATe][?]
      [:FILTER][?]
    [:SELect][?]
  [:PRBSquence][?]
  [:SELect][?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:STATe][?]</td>
<td>[:SOURce]:JITTter:HFRequency:BUNCOrrelate[:STATe][?] on page 163</td>
</tr>
<tr>
<td>[:AMPLitude][?]</td>
<td>[:SOURce]:JITTter:HFRequency:BUNCOrrelate:AMPLitude[?] on page 163</td>
</tr>
<tr>
<td>[:DRATe][?]</td>
<td>[:SOURce]:JITTter:HFRequency:BUNCOrrelate:DRATe[?] on page 164</td>
</tr>
<tr>
<td>[:FILTER][?]</td>
<td>[:SOURce]:JITTter:HFRequency:BUNCOrrelate:FILTER[:SELect][?] on page 164</td>
</tr>
<tr>
<td>[:PRBSquence][?]</td>
<td>[:SOURce]:JITTter:HFRequency:BUNCOrrelate:PRBSquence[:SELect][?] on page 165</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer:HFRequency:BUNCorrelate[:STATe]?[
] Syntax  [:SOURce]:JITTer:HFRequency:BUNCorrelate[:STATe] 'identifier',<ON|OFF>[1|0>
[:SOURce]:JITTer:HFRequency:BUNCorrelate[:STATe]? 'identifier'
Input  'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'
Parameters  <ON|OFF>[1|0]: Enable/disable bounded uncorrelated jitter.
Return Range  1|0
Description Enables/disables the generation of bounded uncorrelated jitter at the specified output.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.
Example  
[:SOURce]:JITTer:HFRequency:BUNCorrelate:AMPLitude?[
] Syntax  [:SOURce]:JITTer:HFRequency:BUNCorrelate:AMPLitude 'identifier',<NRf>
[:SOURce]:JITTer:HFRequency:BUNCorrelate:AMPLitude? 'identifier'
Input  'identifier': 'M*.DataOut1' or 'M*.DataOut2', or 'M*.DataOut'
Parameters  <NRf>: Set bounded uncorrelated jitter amplitude.
Return Range  NRf
Description This command sets the bounded uncorrelated jitter amplitude in seconds (TIME) or unit intervals (UINTerval). The units are set using the [:SOURce]:JITTer:HFRequency:UNIT command.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.
Example  
::JITT:HFR:BUNC 'M2.DataOut2', 400e-12
[:SOURce]:JITTer:HFRequency:BUNCorrelate:DRATe[?] Syntax [:SOURce]:JITTer:HFRequency:BUNCorrelate:DRATe 'identifier', <RATE625|RATE1250|RATE2500> [:SOURce]:JITTer:HFRequency:BUNCorrelate:DRATe? 'identifier' Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut' Parameters <RATE625|RATE1250|RATE2500>: Set the bounded uncorrelated PRBS data rate. Return Range RATE625|RATE1250|RATE2500 Description This command sets the bounded uncorrelated PRBS data rate to 625 MBps, 1250 MBps, or 2500 MBps. This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A. Example :JITT:HFR:BUNC:DRAT 'M2.DataOut2', RATE625

[:SOURce]:JITTer:HFRequency:BUNCorrelate:FILTer[:SELect][?] Syntax [:SOURce]:JITTer:HFRequency:BUNCorrelate:FILTer[:SELect] 'identifier', <LP50|LP100|LP200> [:SOURce]:JITTer:HFRequency:BUNCorrelate:FILTer[:SELect]? 'identifier' Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut' Parameters <LP50|LP100|LP200>: Select the low-pass filter. Return Range LP50|LP100|LP200 Description This command selects the low-pass filter for bounded uncorrelated jitter (50, 100, or 200 MHz). This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A. Example :JITT:HFR:BUNC:FILT 'M2.DataOut2', LP50
[:SOURce]:JITT:HFRequency:BUNCorrelate:PRBSequence[:SElect]?[

Syntax
[:SOURce]:JITT:HFRequency:BUNCorrelate:PRBSequence[:SElect]
 'identifier',
<PRBS7|PRBS8|PRBS9|PRBS10|PRBS11|PRBS15|PRBS23|PRBS31
[:SOURce]:JITT:HFRequency:BUNCorrelate:PRBSequence[:SElect]?[

Input Parameters
'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'
<PRBS7|PRBS8|PRBS9|PRBS10|PRBS11|PRBS15|PRBS23|PRBS31>: Select the PRBS polynomial.

Return Range PRBS7|PRBS8|PRBS9|PRBS10|PRBS11|PRBS15|PRBS23|PRBS31

Description This command selects polynomial of the PRBS for bounded uncorrelated jitter source.
This query returns the current setting.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example :JITT:HFR:BUNC:PRBS 'M2.DataOut2', PRBS7
This subnode has the following SCPI structure:

```
[:SOURce]
  :JITTer
  :HFRequency
  :RANDom
    [:STATe][?]
    :AMPLitude [?]
    :FILTER
      [:LPASs][?]
      :VALue[?]
      :HPASs[?]
      :VALue[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:STATe][?]</td>
<td>[:SOURce]:JITTer:HFRequency:RANDom[:STATe][?] on page 167</td>
</tr>
<tr>
<td>:AMPLitude[?]</td>
<td>[:SOURce]:JITTer:HFRequency:RANDom:AMPLitude[?] on page 167</td>
</tr>
<tr>
<td>:FILTER[:LPASs][?]</td>
<td>[:SOURce]:JITTer:HFRequency:RANDom:FILTER[:LPASs][?] on page 168</td>
</tr>
<tr>
<td>:FILTER[:LPASs]:VALue[?]</td>
<td>[:SOURce]:JITTer:HFRequency:RANDom:FILTER[:LPASs]:VALue[?] on page 168</td>
</tr>
<tr>
<td>:FILTER:HPASs[?]</td>
<td>[:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs[?] on page 169</td>
</tr>
<tr>
<td>:FILTER:HPASs:VALue[?]</td>
<td>[:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs:VALue[?] on page 169</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer:HFRequency:RANDom[:STATe]?]

Syntax
[:SOURce]:JITTer:HFRequency:RANDom[:STATe] 'identifier', <ON|OFF|1|0>

[:SOURce]:JITTer:HFRequency:RANDom[:STATe]? 'identifier'

Input
Identifier: 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'

Parameters
<ON|OFF|1|0>: Enable/disable random jitter source.

Return Range
0|1

Description
This command enables/disables the generation of random jitter.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

This SCPI is also available on M8195A for DataOut1 to 4.

Example
:JITT:HFR:RAND 'M2.DataOut2', on

[:SOURce]:JITTer:HFRequency:RANDom:AMPLitude[?]

Syntax
[:SOURce]:JITTer:HFRequency:RANDom:AMPLitude 'identifier', <NRf>

[:SOURce]:JITTer:HFRequency:RANDom:AMPLitude? 'identifier'

Input
Identifier: 'M*.DataOut1' or 'M*.DataOut2', or 'M*.DataOut'

Parameters
<NRF>: Set random jitter amplitude.

Return Range
0 uUI to 13.08 mUI or 0 ps to 78.57 ps

Description
This command sets the root mean square (rms) RJ amplitude in seconds or unit intervals based on the selected amplitude unit. The units are set using the [:SOURce]:JITTer:HFRequency:UNIT command.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

This SCPI is also available on M8195A for DataOut1 to 4.

Example
:JITT:HFR:RAND:AMPL 'M2.DataOut2', 0.013
[:SOURce]:JITT:HFR:RAND:FIILT[:LPAS]:? Syntax [:SOURce]:JITT:HFR:RAND:FIILT[:LPAS]: 'identifier',<OFF|LP100|LP500> [:SOURce]:JITT:HFR:RAND:FIILT[:LPAS]? 'identifier' Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut' Parameters <OFF|LP100|LP500>: Enable/disable low-pass filter. Return Range OFF|LP100|LP500 Description This command enables/disables the low-pass filter for random jitter. LP100 enables a 100 MHz low-pass filter; LP500 enables a 500 MHz low-pass filter. This query returns the current settings. This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A. Example :JITT:HFR:RAND:FIILT 'M1.DataOut2', LP100

[:SOURce]:JITT:HFR:RAND:FIILT[:LPAS]:VALue? Syntax [:SOURce]:JITT:HFR:RAND:FIILT[:LPAS]:VALue 'identifier',<NRf> [:SOURce]:JITT:HFR:RAND:FIILT[:LPAS]:VALue? 'identifier' Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', 'M*.DataOut4', or 'M*.DataOut' Parameters <NRf> Return Range 1 MHz to 10 GHz The limit is dynamic, it depends on the high-pass setting. Description This command controls the low pass filter of the random jitter source. This query returns the current settings. This SCPI is applicable for M8195A and M8196A. Example :JITT:HFR:RAND:FIILT:LPAss:VALue 'M1.DataOut2',1e6
### [:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs

**Syntax**  
[:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs 'identifier'  
<OFF|HP10>  
[:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs? 'identifier'

**Input Parameters**  
- 'identifier': 'M*.DataOut1' or 'M*.DataOut2', or 'M*.DataOut'

**Return Range**  
OFF|HP10

**Description**  
This command enables/disables the high-pass filter for random jitter.  
HP10 enables a 10 MHz high-pass filter.  
This query returns the current settings.  
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

**Example**  

### [:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs:VALue

**Syntax**  
[:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs 'identifier' <NRf>  
[:SOURce]:JITTer:HFRequency:RANDom:FILTER:HPASs:VALue? 'identifier'

**Input Parameters**  
- 'identifier': 'M*.DataOut1' or 'M*.DataOut2', or 'M*.DataOut'

**Return Range**  
1 kHz to 10 MHz  
The limit is dynamic, it depends on the low-pass setting.

**Description**  
This command controls the high pass filter value of the random jitter source.  
This SCPI is applicable for M8195A and M8196A.  
This query returns the current settings.

**Example**  
Spectrally distributed random jitter is composed of two jitter sources: low frequency jitter and high frequency jitter. It is characterized by the amplitudes of the low and the high frequency jitter part.

This subnode has the following SCPI structure:

![SCPI Structure Diagram]

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:STATe]?</td>
<td>[:SOURce]:JITTer:HFRequency:SPECtrally[:STATe]? on page 171</td>
</tr>
<tr>
<td>:AMPLitude(*)?</td>
<td>[:SOURce]:JITTer:HFRequency:SPECtrally:AMPLitude(*)? on page 171</td>
</tr>
<tr>
<td>:FILTER[LPASs]:STATe]?</td>
<td>[:SOURce]:JITTer:HFRequency:SPECtrally:FILTER[LPASs]:STATe]? on page 172</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer:HFRequency:SPECtrally[:STATe]?[?]

**Syntax**

[:SOURce]:JITTer:HFRequency:SPECtrally[:STATe] 'identifier', 'ON|OFF|1|0'

[:SOURce]:JITTer:HFRequency:SPECtrally[:STATe]? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'
- '<ON|OFF|1|0>': Enable/disable spectrally distributed jitter.

**Return Range**

0|1

**Description**

This command enables/disables the generation of spectrally distributed random jitter.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

**Example**

:JITT:HFR:SPEC 'M2.DataOut2', off

[:SOURce]:JITTer:HFRequency:SPECtrally:AMPLitude(*)[?]

**Syntax**

[:SOURce]:JITTer:HFRequency:SPECtrally:AMPLitude(*) 'identifier', <NRf>

[:SOURce]:JITTer:HFRequency:SPECtrally:AMPLitude(*)? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'
- '<NRf>': Set spectrally distributed jitter amplitude.

**Return Range**

0 mUI to 78.71 mUI

**Description**

This command sets the spectrally distributed jitter amplitude in unit intervals or time.

A suffix attached to the :AMPLitude(*) command is used to address two jitter amplitudes. The :AMPL1 <NR3> command changes the level of the internal RJLF resource and the :AMPL2 <NR3> command changes the level of the internal RJHF resource. Both levels are specified in rms.

The units are set using the [:SOURce]:JITTer:HFRequency:UNIT command.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

**Example**

:JITT:HFR:SPEC:AMPL2 'M2.DataOut2', 0.013
[:SOURce]:JITTER:HFRequency:SPECTrally:FILTer[:LPASs][:STATE]?

**Syntax**

[:SOURce]:JITTER:HFRequency:SPECTrally:FILTer[:LPASs][:STATE] 

'identifier', <ON|OFF|1|0>

[:SOURce]:JITTER:HFRequency:SPECTrally:FILTer[:LPASs][:STATE]?

'identifier'

**Input Parameters**

'identifier': 'M*.DataOut1' or 'M*.DataOut2'

<ON|OFF|1|0>: Enable/disable low-pass filter.

**Return Range**

1|0

**Description**

Enables/disables the 100 MHz low-pass filter for spectrally distributed random jitter.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

**Example**

:JITT:HFR:SPEC:FILT 'M2.DataOut2', off
This subnode has the following SCPI structure:

```
[:SOURce]:JITTER:LFRequency
  [:UNIT?]     [:SOURce]:JITTER:LFRequency:UNIT[?] on page 174
  :PERiodic   [:SOURce]:JITTER:LFRequency:PERiodic Subnode on page 175
  :RSSClocking [:SOURce]:JITTER:LFRequency:RSSClocking Subnode on page 179
```

This subnode has the following commands:

**Table 41**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:UNIT[?]</td>
<td>[:SOURce]:JITTER:LFRequency:UNIT[?] on page 174</td>
</tr>
<tr>
<td>:PERiodic</td>
<td>[:SOURce]:JITTER:LFRequency:PERiodic Subnode on page 175</td>
</tr>
<tr>
<td>:RSSClocking</td>
<td>[:SOURce]:JITTER:LFRequency:RSSClocking Subnode on page 179</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTER:LFrequency:UNIT[?]

Syntax

[:SOURce]:JITTER:LFrequency:UNIT 'identifier', <UINTerval|TIME>
[:SOURce]:JITTER:LFrequency:UNIT? 'identifier'

Input Parameters

'identifier': 'M1.ClkOut', 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Return Range

UINT|TIME

Description

Specifies whether the jitter parameters are to be specified/returned in seconds (TIME) or unit intervals (UINTerval).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example

:JITT:LFR:UNIT 'M1.DataOut2',TIME
This subnode has the following SCPI structure:

```
[:SOURce]
  :JITTer
  :LFRequency
    :PERiodic1
      [:STATe][?]
      [:AMPLitude][?]
      [:FREQuency][?]
    :PERiodic2
      [:STATe][?]
      [:AMPLitude][?]
      [:FREQuency][?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERiodic1[:STATe][?]</td>
<td>[:SOURce]:JITTer:LFRequency:PERiodic1[:STATe][?] on page 176</td>
<td></td>
</tr>
<tr>
<td>PERiodic1:AMPLitude[?]</td>
<td>[:SOURce]:JITTer:LFRequency:PERiodic1:AMPLitude[?] on page 176</td>
<td></td>
</tr>
<tr>
<td>PERiodic1:FREQuency[?]</td>
<td>[:SOURce]:JITTer:LFRequency:PERiodic1:FREQuency[?] on page 176</td>
<td></td>
</tr>
<tr>
<td>PERiodic2[:STATe][?]</td>
<td>[:SOURce]:JITTer:LFRequency:PERiodic2[:STATe][?] on page 177</td>
<td></td>
</tr>
<tr>
<td>PERiodic2:AMPLitude[?]</td>
<td>[:SOURce]:JITTer:LFRequency:PERiodic2:AMPLitude[?] on page 177</td>
<td></td>
</tr>
<tr>
<td>PERiodic2:FREQuency[?]</td>
<td>[:SOURce]:JITTer:LFRequency:PERiodic2:FREQuency[?] on page 178</td>
<td></td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer:LFRequency:PERiodic1[:STATe][?] Syntax [:SOURce]:JITTer:LFRequency:PERiodic1[:STATe] 'identifier', <ON|OFF|1|0> [:SOURce]:JITTer:LFRequency:PERiodic1[:STATe]? 'identifier'

Input Parameters

'identifier': 'M1.ClkOut', 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Return Range  <ON|OFF|1|0>: Enable/disable the low frequency periodic 1 jitter source.

Description This command enables/disables the low frequency periodic 1 jitter source. This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example :JITT:LFR:PER1 'M1.DataOut', ON


Input Parameters

'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'

Return Range  <NRf>: Set periodic 1 low frequency jitter amplitude.

Description This command sets the amplitude of the periodic 1 low frequency jitter in unit intervals or time for the specified output. The units are set using the [:SOURce]:JITTer:LFRequency:UNIT command. This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example :JITT:LFR:PER1:AMPL 'M2.DataOut2', 1


Input Parameters

'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'

Return Range  <NRf>: Set periodic 1 jitter frequency.
Description
This command sets the frequency of the periodic 1 jitter. Acceptable units include Hz, kHz, MHz and exponents (for example, 1E3 is the same as 1,000 Hz).
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example
:JITT:LFR:PER1:FREQ 'M2.DataOut2', 1e6

[:SOURce]:JITTer:LFRequency:PERiodic2[:STATe]?

Syntax
[:SOURce]:JITTer:LFRequency:PERiodic2[:STATe] 'identifier', <ON|OFF|1|0>
[:SOURce]:JITTer:LFRequency:PERiodic2[:STATe]? 'identifier'

Input Parameters
'identifier': 'M1.ClkOut', 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'
<ON|OFF|1|0>: Enable/disable the low frequency periodic 2 jitter source.

Return Range 1|0

Description
This command enables/disables the low frequency periodic 2 jitter source.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example
:JITT:LFR:PER2 'M1.DataOut', ON

[:SOURce]:JITTer:LFRequency:PERiodic2:AMPLitude[

Syntax
[:SOURce]:JITTer:LFRequency:PERiodic2:AMPLitude 'identifier', <NRf>
[:SOURce]:JITTer:LFRequency:PERiodic2:AMPLitude? 'identifier'

Input Parameters
'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'
<NRf>: Set periodic 2 low frequency jitter amplitude.

Return Range NRf

Description
This command sets the amplitude of the periodic 2 low frequency jitter in unit intervals or time for the specified output. The units are set using the [:SOURce]:JITTer:LFRequency:UNIT command.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example
:JITT:LFR:PER2:AMPL 'M2.DataOut2', 1
[:SOURce]:JITT:LFRequency:PERiodic2:FREQuency[?]

Syntax

[:SOURce]:JITT:LFRequency:PERiodic2:FREQuency 'identifier', <NRf>
[:SOURce]:JITT:LFRequency:PERiodic2:FREQuency? 'identifier'

Input

'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'

Parameters

<NRf>: Set periodic 2 jitter frequency.

Return Range

NRf

Description

This command sets the frequency of the periodic 2 jitter. Acceptable units include Hz, kHz, MHz and exponents (for example, 1E3 is the same as 1,000 Hz).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example

:JITT:LFR:PER2:FREQ 'M2.DataOut2', 1e6
This subnode has the following SCPI structure:

```
[:SOURce]:JITTer:LFRRequency:RSSClocking
```

This subnode has the following commands:

### Table 43

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:STATe]?</td>
<td>[:SOURce]:JITTer:LFRRequency:RSSClocking[:STATe]? on page 179</td>
</tr>
<tr>
<td>::AMPLitude?</td>
<td>[:SOURce]:JITTer:LFRRequency:RSSClocking::AMPLitude? on page 180</td>
</tr>
<tr>
<td>::FREQuency?</td>
<td>[:SOURce]:JITTer:LFRRequency:RSSClocking::FREQuency? on page 180</td>
</tr>
</tbody>
</table>

[:SOURce]:JITTer:LFRRequency:RSSClocking[:STATe]?]

**Syntax**

```
[:SOURce]:JITTer:LFRRequency:RSSClocking[:STATe]? 'identifier', <ON|OFF>[1|0>
```

**Input Parameters**

- ‘identifier’: 'M1.ClkOut', 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'
- <ON|OFF>[1|0>: Enable/disable residual spread spectrum clocking.

**Return Range**

1|0

**Description**

This command enables/disables the residual spread spectrum clocking (rSSC).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8145A.

**Example**

```
:JITT:LFR:RSSC 'M2.DataOut2', on
```
[:SOURce]:JITT:LFRequency:RSSClocking:AMPLitude[?]

Syntax
[:SOURce]:JITT:LFRequency:RSSClocking:AMPLitude 'identifier', <NRf>
[:SOURce]:JITT:LFRequency:RSSClocking:AMPLitude? 'identifier'

Input
Parameters

'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'

Input
Parameters

<Nrf>: Set residual spread spectrum clocking amplitude.

Description
This command sets the amplitude of the residual spread spectrum clocking (rSSC). The units are set using the [:SOURce]:JITT:LFRequency:UNIT command.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8145A.

Example
:JITT:LFR:RSSC:AMPL 'M2.DataOut2', 0.016

[:SOURce]:JITT:LFRequency:RSSClocking:FREQuency[?]

Syntax
[:SOURce]:JITT:LFRequency:RSSClocking:FREQuency 'identifier', <NRf>
[:SOURce]:JITT:LFRequency:RSSClocking:FREQuency? 'identifier'

Input
Parameters

'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M1.ClkOut', or 'M*.DataOut'

Input
Parameters

<Nrf>: Set residual spread spectrum clocking frequency.

Return Range
10 kHz to 100 kHz

Description
This command sets the frequency of the residual spread spectrum clocking (rSSC). Acceptable units include Hz, kHz and exponents (for example, 10E3 is the same as 10,000 Hz).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8145A.

Example
:JITT:LFR:RSSC:FREQ 'M2.DataOut1', 10khz
[:SOURce]:JITTer:SWEep Subnode

This subnode has the following SCPI structure:

```
[:SOURce]
  :JITTer
  :SWEep
    [:STATe][?]
    :FREQuency
    :STARt[?]
    :STOP[?]
    :TIME[?]
    :STEP
      :DISTance[?]
      [:VALue]?
    :AMPLitude
    :MODE[?]
    [:VALue]?
    :PLOT
      :INPut?
      [:VALue]?
    DATA
      :FILE[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:STATe][?]</td>
<td>[:SOURce]:JITTer:SWEep[:STATe][?] on page 182</td>
</tr>
<tr>
<td>:FREQuency:STARt[?]</td>
<td>[:SOURce]:JITTer:SWEep:FREQuency:STARt[?] on page 182</td>
</tr>
<tr>
<td>:FREQuency:STOP[?]</td>
<td>[:SOURce]:JITTer:SWEep:FREQuency:STOP[?] on page 183</td>
</tr>
<tr>
<td>:TIME[?]</td>
<td>[:SOURce]:JITTer:SWEep:TIME[?] on page 183</td>
</tr>
<tr>
<td>:STEP[:VALue][?]</td>
<td>[:SOURce]:JITTer:SWEep:STEP[:VALue][?] on page 184</td>
</tr>
<tr>
<td>:STEP:DISTance[?]</td>
<td>[:SOURce]:JITTer:SWEep:STEP:DISTance[?] on page 184</td>
</tr>
</tbody>
</table>
[:SOURce]:JITTer:SWEep[:STATe][?]
Syntax [:SOURce]:JITTer:SWEep[:STATe] 'identifier', <ON|OFF|1|0>
[:SOURce]:JITTer:SWEep[:STATe]? 'identifier'
Input 'identifier': 'M*.DataOut1' or 'M*.DataOut2'
Parameters <ON|OFF|1|0>: Enable/disable periodic jitter sweep.
Return Range 1|0
Description This command enables/disables the jitter sweep. This SCPI is applicable for M8041A and M8051A.
Example :JITT:SWE 'M1.DataOut2', on

[:SOURce]:JITTer:SWEep:FREQuency:STARt[?]
Syntax [:SOURce]:JITTer:SWEep:FREQuency:STARt 'identifier', <NRf>
[:SOURce]:JITTer:SWEep:FREQuency:STARt? 'identifier'
Input 'identifier': 'M*.DataOut1' or 'M*.DataOut2'
Parameters <NRf>: Set the start frequency.
Return Range 1 kHz to 500 MHz
Description This command defines the start frequency of the jitter sweep. The start frequency must be lower than the stop frequency and the range should be in accordance with the waveform. Acceptable units include Hz, kHz, MHz, GHz and exponents (for example, 1E3 is the same as 1,000 Hz).
This SCPI is applicable for M8041A and M8051A.


`:SOURce:JITTer:SWEep:FREQuency:STOP[?]

Syntax: `[:SOURce]:JITTer:SWEep:FREQuency:STOP 'identifier', <NRf>`

Input Parameters:

- `'identifier'`: 'M*.DataOut1' or 'M*.DataOut2'
- `<NRf>`: Set the stop frequency.

Return Range: 1 kHz to 500 MHz

Description:

This command defines the stop frequency of the jitter sweep. The stop frequency must be higher than the start frequency and the range should be in accordance with the waveform. Acceptable units include Hz, kHz, MHz, GHz and exponents (for example, 1E3 is the same as 1,000 Hz).

This SCPI is applicable for M8041A and M8051A.


`:SOURce:JITTer:SWEep:TIME[?]

Syntax: `[:SOURce]:JITTer:SWEep:TIME 'identifier', <NRf>`

Input Parameters:

- `'identifier'`: 'M*.DataOut1' or 'M*.DataOut2'
- `<NRf>`: Set duration of sweep.

Return Range: 100 ms to 1.2 ks

Description:

This command defines the time duration of the jitter sweep. Acceptable units include ms (milliseconds), s (seconds), ks (kiloseconds) and exponents (for example, 1E3 is the same as 1,000 seconds).

This SCPI is applicable for M8041A and M8051A.

Example: `:JITT:SWE:TIME 'M1.DataOut2', 100ms`
[:SOURce]:JITTer:SWEep:STEP:[VALue][?] Syntax

[:SOURce]:JITTer:SWEep:STEP:[VALue] 'identifier', <NRf>
[:SOURce]:JITTer:SWEep:STEP:[VALue]? 'identifier'

Input

Parameters 'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Return Range 2 to 100

Description This command defines the number of steps to fulfill a complete jitter sweep.

This SCPI is applicable for M8041A and M8051A.

Example :JITT:SWE:STEP 'M1.DataOut2', 50

[:SOURce]:JITTer:SWEep:STEP:DISTance[?] Syntax

[:SOURce]:JITTer:SWEep:STEP :DISTance 'identifier', <EQUidistant|AUTomatic>
[:SOURce]:JITTer:SWEep:STEP :DISTance? 'identifier'

Input

Parameters 'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Return Range EQU|AUTO

Description The command defines whether the frequency steps are log equidistant (EQUidistant) along the periodic jitter curve or a frequency step matches a corner frequency on the jitter curve.

This SCPI is applicable for M8041A and M8051A.

Example :JITT:SWE:STEP:DIST 'M1.DataOut2', AUTO

[:SOURce]:JITTer:SWEep:AMPLitude[:VALue][?]

Syntax [:SOURce]:JITTer:SWEep:AMPLitude[:VALue] 'identifier', <NR3>
[:SOURce]:JITTer:SWEep:AMPLitude[:VALue]? 'identifier'

Input

Parameters <NR3>: Specify constant jitter amplitude of the jitter sweep.

Return Range 0 mUI to 1.102 UI
This command sets the constant jitter amplitude of the jitter sweep. If CONStant jitter amplitude is selected as the jitter sweep mode, this command is used to define the value of the constant jitter amplitude of the jitter sweep. Acceptable units include mUI, UI and exponents (for example, 100e-3 is the same as 100 mUI).

This SCPI is applicable for M8041A and M8051A.

Example :
```
JITT:SWE:AMPL 'M1.DataOut2', 1UI
```

Syntax
```
[:SOURce]:JITTer:SWEep:AMPLitude:MODE['identifier', <CONStant|VARiable>]
```

Input
- `identifier`: 'M*.DataOut1' or 'M*.DataOut2'
- `<CONStant|VARiable>`: Specify constant or user defined jitter level.

Return Range
- CONS|VAR

Description
The command defines whether a sweep is defined with CONStant jitter level (amplitude) or with the user defined VARiable jitter level (amplitude). This SCPI is applicable for M8041A and M8051A.

Example :
```
JITT:SWE:AMPL:MODE 'M1.DataOut2', CONS
```

Syntax
```
[:SOURce]:JITTer:SWEep:PLOT[:VALue]['identifier']
```

Input
- `identifier`: 'M*.DataOut1' or 'M*.DataOut2'

Return Range
- <frequency/jitter amplitude value pairs>

Description
This query returns a comma separated list (specified as an expression) of frequency/jitter amplitude value pairs. This list of values represents the 'real' steps performed by the jitter sweep. This SCPI is applicable for M8041A and M8051A.

Example :
```
JITT:SWE:PLOT? 'M1.DataOut2'
```
[::SOURce::JITTer::SWEep::PLOT::INPut?]

Syntax  
[::SOURce::JITTer::SWEep::PLOT::INPut? 'identifier']

Input  
'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Parameters

Description  
This query returns a comma separated list (specified as an expression) of frequency/jitter amplitude value pairs. This list of values represents the value pairs of the corner frequencies only.

This SCPI is applicable for M8041A and M8051A.

Example  
::JITT:SWE:PLOT:INP? 'M1.DataOut2'

[::SOURce::JITTer::SWEep::DATA::FILE[?]]

Syntax  
[::SOURce::JITTer::SWEep::DATA::FILE 'identifier', '<FileName>'

[::SOURce::JITTer::SWEep::DATA::FILE? 'identifier']

Input  
'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Parameters

'<FileName>': Specify path to a data file.

Description
Every point of the periodic jitter sweep curve can be specified by a group of 2 value pairs Frequency, JitterLevel, Frequency, JitterLevel, etc. This command specifies the location of the file to be loaded within the workspace.

Jitter sweep profiles can be stored in three different areas:
1. Local to current setting ('current/')
2. Shared between settings ('shared/')
3. Factory supplied standard sweep profiles ('factory/').
These sweep profiles are read only and cannot be modified.

Below these root nodes there is a folder structure using '/' as separation character. So, a complete file name might be something like 'factory/PCIe_8G_CC.jcs', or 'shared/subfolder/mySweepProfile.jcs'.

This SCPI is applicable for M8041A and M8051A.

Example  
::JITT:SWE:DATA:FILE 'M1.DataOut2', 'factory/PCIe_8G_CC.jcs'
[:SOURce]:SSCLocking Subnode

Spread Spectrum Clocking (SSCL) is used for reducing the peak electromagnetic radiation. The clock out signal and data out stream are modulated with a repetitive low-frequency waveform.

This subnode has the following SCPI structure:

```
[:SOURce]
  ::SSCLocking
    ::STATe[?]
    ::GLOBal::STATe[?]
    ::DEViation[?]
    ::TYPE[?]
    ::PROFile[?]
    ::SHAPE[?]
    ::FREQuency[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>::STATe[?]</td>
<td>[:SOURce]:SSCLocking::STATe[?] on page 188</td>
</tr>
<tr>
<td>GLOBal::STATe[?]</td>
<td>[:SOURce]:SSCLocking::GLOBal::STATe[?] on page 188</td>
</tr>
<tr>
<td>::DEViation[?]</td>
<td>[:SOURce]:SSCLocking::DEViation::VALue[?]? on page 189</td>
</tr>
<tr>
<td>::TYPE[?]</td>
<td>[:SOURce]:SSCLocking::TYPE[?]? on page 190</td>
</tr>
<tr>
<td>::PROFile[?]</td>
<td>[:SOURce]:SSCLocking::PROFile[?]? on page 191</td>
</tr>
<tr>
<td>::SHAPE[?]</td>
<td>[:SOURce]:SSCLocking::SHAPE[?]? on page 192</td>
</tr>
<tr>
<td>::FREQuency[?]</td>
<td>[:SOURce]:SSCLocking::FREQuency[?]? on page 192</td>
</tr>
</tbody>
</table>
[:SOURce]:SSCLocking[:STATe][?] Syntax 
[:SOURce]:SSCLocking[:STATe] 'identifier', <ON|OFF|1|0> 
[:SOURce]:SSCLocking[:STATe]? 'identifier' 
Input 'identifier': 'M1.ClkGen' 
Parameters <ON|OFF|1|0>: Enable/disable spread spectrum clocking. 
Return Range 1|0 
Description This command enables/disables spread spectrum clocking (SSC) state. 
The SSC mode can be only enabled if Global SSC state is activated. 
This SCPI is applicable for M8041A, M8195A, M8196A and M8045A. 
Example The below mentioned SCPI command turns ON the SSC state: 
:SCL 'M1.ClkGen', ON 
The below mentioned SCPI command queries the SSC: 
:SSCL? 'M1.ClkGen' 
1

:SOURce]:SSCLocking:GLOBal[:STATe][?] Syntax [:SOURce]:SSCLocking:GLOBal[:STATe] 'identifier', <ON|OFF|1|0> [:SOURce]:SSCLocking:GLOBal[:STATe]? 'identifier' 
Input 'identifier': 'M1.System' 
Parameters <ON|OFF|1|0>: Enable/disable spread spectrum clocking. 
Return Range 1|0 
Description This command turns the spread spectrum clocking (SSC) mode globally on or off. 
This means that SSC will be only enable for a specific module if the module specific SSC switch and the global SSC switch are both turned on, otherwise SSC is disabled. 
[:SOURce]:SSCLocking:DEViation[:VALue]:?

Syntax 
[:SOURce]:SSCLocking:DEViation[:VALue] 'identifier', <NRf>
[:SOURce]:SSCLocking:DEViation[:VALue]? 'identifier'

Input 
'identifier': 'M1.ClkGen'

Parameters 
<NRf>: Set spread spectrum clocking deviation.

Return Range 
0% to 1%

Description 
This command sets the spread spectrum clocking (SSC) deviation in percent. The effective deviation is ±0.5% (or 1% peak-to-peak).

This SCPI is applicable for M8041A, M8195A, M8196A and M8045A.

Example 
The below mentioned SCPI command sets the spread spectrum clocking (SSC) deviation to '0.2' percent for M8041A/M8045A:
SSCL:DEV 'M1.ClkGen', 0.2

The below mentioned SCPI command queries the spread spectrum clocking (SSC) deviation for M8041A/M8045A:
SSCL:DEV? 'M1.ClkGen'
0.2

[:SOURce]:SSCLocking:DEViation:UP[:?]

Syntax 
[:SOURce]:SSCLocking:DEViation:UP 'identifier', <NRf>
[:SOURce]:SSCLocking:DEViation:UP? 'identifier'

Input 
'identifier': 'M1.ClkGen'

Parameters 
<NRf>: Set peak up spread spectrum clocking deviation.

Description 
This command sets the up deviation from the currently set clock rate. Value range is from 0.0 % to +1.0 %. Sending this command will automatically change the SSC type to ASYMmetric.

This SCPI is applicable for M8041A and M8045A.

Example 
The below mentioned SCPI command sets the up deviation to '0.25' percent for M8041A/M8045A:
SSCL:DEV:UP 'M1.ClkGen', 0.25
The below mentioned SCPI command queries the up deviation for M8041A/M8045A:

SSCL:DEV:UP? 'M1.ClkGen'

0.25

[:SOURce]:SSCLocking:DEViation:DOWN[?]

Syntax

[:SOURce]:SSCLocking:DEViation:DOWN 'identifier', <NRf>
[:SOURce]:SSCLocking:DEViation:DOWN? 'identifier'

Input Parameters

'identifier': 'M1.ClkGen'

<NRf>: Set spread spectrum clocking deviation.

Description

This command sets the down deviation from the currently set clock rate. Value range is from -1.0 % to 0.0 %. Sending this command will automatically change the SSC type to ASYMmetric.

This SCPI is applicable for M8041A and M8045A.

Example

The below mentioned SCPI command sets down deviation to '-0.45' percent for M8041A/M8045A:

SSCL:DEV 'M1.ClkGen', -0.45

The below mentioned SCPI command queries the down deviation for M8041A/M8045A:

SSCL:DEV? 'M1.ClkGen'

-0.45

[:SOURce]:SSCLocking:TYPE[?]

Syntax

[:SOURce]:SSCLocking:TYPE 'identifier', <DOWNspread|UPSPread|CENTerspread|ASYMmetric>
[:SOURce]:SSCLocking:TYPE? 'identifier'

Input Parameters

'identifier': 'M1.ClkGen'

<DOWNspread|UPSPread|CENTerspread|ASYMmetric>: Set spread spectrum clocking type.

Return Range

DOWN|UPSP|CENT|ASYM
Description

This command sets method used to modulate the clock frequency and data. Choose between center spread, down spread, up spread or asymmetric. For SSC type as ASYMmetric, you have to individually specify up and down deviation. For details, see [:SOURce]:SSCLocking:DEViation:UP[?] on page 189 and [:SOURce]:SSCLocking:DEViation:DOWN[?] on page 190.

This SCPI is applicable for M8041A, M8095A and M8045A.
It is also applicable for M8195A but only in query mode.

Example

The below mentioned SCPI command sets the method used to modulate clock frequency and data to 'DOWN' for M8041A/M8045A:

:SSCL:TYPE 'M1.ClkGen', DOWN

The below mentioned SCPI command queries the method used to modulate clock frequency and data for M8041A/M8045A:

:SSCL:TYPE? 'M1.ClkGen'
DOWN

[:SOURce]:SSCLocking:PROFile[?]

Syntax

[:SOURce]:SSCLocking:PROFile 'identifier', <TRIangular|ARBitrary>
[:SOURce]:SSCLocking:PROFile? 'identifier'

Input

'identifier': 'M1.ClkGen'

Parameters

<TRIangular|ARBitrary>: Set spread spectrum clocking profile.

Return Range

TR|ARB

Description

This command sets up the spread spectrum clocking (SSC) using a triangular or arbitrary profile.

This SCPI is applicable for M8041A and M8045A.
It is also applicable for M8195A but only in query mode.

Example

:SSCL:PROF 'M1.ClkGen', TRI
`:SOURce`:SSCLocking:SHAPe[?]

Syntax

`:SOURce`:SSCLocking:SHAPe 'identifier', `<Filename>`

`:SOURce`:SSCLocking:SHAPe? 'identifier'

Input

'identifier': 'M1.ClkGen'

Parameters

`<Filename>`: Set up spread spectrum clocking shape specified in 'Filename'.

Return Range

`<Filename>`

Description

This command sets up the spread spectrum clocking (SSC) using the specified shape in "Filename". A simple text file contains the data points which define the arbitrary waveform of the SSC profile.

This SCPI is applicable for M8041A and M8045A.

Example


`:SOURce`:SSCLocking:FREQuency[?]

Syntax

`:SOURce`:SSCLocking:FREQuency 'identifier', `<NRf>`

`:SOURce`:SSCLocking:FREQuency? 'identifier'

Input

'identifier': 'M1.ClkGen'

Parameters

`<NRf>`: Set spread spectrum clocking frequency.

Return Range

100 Hz to 200 kHz

Description

This command sets the spread spectrum clocking (SSC) frequency in hertz. Acceptable units include Hz, kHz and exponents (for example, 1E3 is the same as 1 kHz).

This SCPI is applicable for M8041A, M8195A, M8196A and M8045A.

Example

`:SSCL:FREQ 'M1.ClkGen', 1khz`
This subnode has the following SCPI structure:

```
[:SOURce]:INTerference
    [:LEVel]
    -:HFRequency[:SOURce]
    -:LFRquency[:SOURce]
    -:CMODE
        -[:STATe][?]
        -:AMPLitude[?]
        -:SOURce[?]
        -:GAIN[?]
    -:DMODE
        -[:STATe][?]
        -:AMPLitude[?]
        -:SOURce[?]
        -:GAIN[?]
    -:ISYMbol
        -[:STATe][?]
        -:MODE[?]
        -:FREQuency{1:2}[?]
        -:ILOSs{1:2}[?]
        -:PRESet[?]
```
This subnode has the following commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description under</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:LEV]e[:FRequency][:SOURce]?</td>
<td>[:SOURce]:INTerference[:LEV]:FRequency[:SOURce]?</td>
<td>195</td>
</tr>
<tr>
<td>[:LEV]e[:FRequency][:SOURce]?</td>
<td>[:SOURce]:INTerference[:LEV]:FRequency[:SOURce]?</td>
<td>195</td>
</tr>
<tr>
<td>[:LEV]e[:CMODE][:STATe]?</td>
<td>[:SOURce]:INTerference[:LEV]:CMODE[:STATe]?</td>
<td>195</td>
</tr>
<tr>
<td>[:LEV]e[:CMODE]:AMPLitude?</td>
<td>[:SOURce]:INTerference[:LEV]:CMODE:AMPLitude?</td>
<td>196</td>
</tr>
<tr>
<td>[:LEV]e[:CMODE][:SOURce]?</td>
<td>[:SOURce]:INTerference[:LEV]:CMODE:SOURce?</td>
<td>196</td>
</tr>
<tr>
<td>[:LEV]e[:CMODE]:GAIN?</td>
<td>[:SOURce]:INTerference[:LEV]:CMODE:GAIN?</td>
<td>197</td>
</tr>
<tr>
<td>[:LEV]e[:DMODE][:STATe]?</td>
<td>[:SOURce]:INTerference[:LEV]:DMODE[:STATe]?</td>
<td>197</td>
</tr>
<tr>
<td>[:LEV]e[:DMODE]:AMPLitude?</td>
<td>[:SOURce]:INTerference[:LEV]:DMODE:AMPLitude?</td>
<td>197</td>
</tr>
<tr>
<td>[:LEV]e[:DMODE][:SOURce]?</td>
<td>[:SOURce]:INTerference[:LEV]:DMODE:SOURce?</td>
<td>198</td>
</tr>
<tr>
<td>[:LEV]e[:DMODE]:GAIN?</td>
<td>[:SOURce]:INTerference[:LEV]:DMODE:GAIN?</td>
<td>198</td>
</tr>
<tr>
<td>:ISYMbol[:STATe]?</td>
<td>[:SOURce]:INTerference:ISYMbol[:STATe]?</td>
<td>199</td>
</tr>
<tr>
<td>:ISYMbol[:MODE]?</td>
<td>[:SOURce]:INTerference:ISYMbol[:MODE]?</td>
<td>199</td>
</tr>
<tr>
<td>:ISYMbol[:FREquency{1:2}]?</td>
<td>[:SOURce]:INTerference:ISYMbol[:FREquency{1:2}]?</td>
<td>199</td>
</tr>
<tr>
<td>:ISYMbol[:ILOSs{1:2}]?</td>
<td>[:SOURce]:INTerference:ISYMbol[:ILOSs{1:2}]?</td>
<td>200</td>
</tr>
<tr>
<td>:ISYMbol[:PRESet]?</td>
<td>[:SOURce]:INTerference:ISYMbol[:PRESet]?</td>
<td>200</td>
</tr>
</tbody>
</table>
[:SOURce]:INTerference[:LEVEL]:HFRequency[:SOURce]?[:?]

Syntax [:SOURce]:INTerference[:LEVEL]:HFRequency[:SOURce] 'identifier', <NRf> [:SOURce]:INTerference[:LEVEL]:HFRequency[:SOURce]? 'identifier'

Input 'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'

Parameters <NRf>: Set frequency of HF generator.

Return Range 1 GHz to 6 GHz

Description This command sets the frequency of the high frequency generator. Acceptable units include MHz, GHz and exponents (for example, 2E9 is the same as 2 GHz).

This SCPI is applicable for M8041A and M8051A.

Example :INT:HFR 'M2.DataOut1', 2.0e9

[:SOURce]:INTerference[:LEVEL]:LFRequency[:SOURce]?[:?]

Syntax [:SOURce]:INTerference[:LEVEL]:LFRequency[:SOURce] 'identifier', <NRf> [:SOURce]:INTerference[:LEVEL]:LFRequency[:SOURce]? 'identifier'

Input 'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'

Parameters <NRf>: Set frequency of LF generator.

Return Range 100 Hz to 1 GHz

Description This command sets the frequency of the low frequency generator. Acceptable units include Hz, kHz, MHz, GHz and exponents (for example, 1E9 is the same as 1 GHz).

This SCPI is applicable for M8041A and M8051A.

Example :INT:LFR 'M2.DataOut1', 1.0e9

[:SOURce]:INTerference[:LEVEL]:CMODe[:STATe]?[:?]

Syntax [:SOURce]:INTerference[:LEVEL]:CMODe[:STATe] 'identifier', <0|1|ON|OFF> [:SOURce]:INTerference[:LEVEL]:CMODe[:STATe]? 'identifier'

Input 'identifier': 'M*.DATAOUT1', 'M*.DataOut2', or 'M*.DataOut'

Parameters <0|1|ON|OFF>: Turn common mode interference on/off.

Return Range 0|1
Description  This command turns common mode interference on/off.
This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

Example  :INT:CMOD 'M2.DataOut1', on

[:SOURce]:INterference[:LEVel]:CMODE:AMPLitude[?

Syntax  [:SOURce]:INterference[:LEVel]:CMODE:AMPLitude 'identifier', <NRf>
[:SOURce]:INterference[:LEVel]:CMODE:AMPLitude? 'identifier'

Input  'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'

Parameters  <NRf>: Set common mode interference amplitude.

Return Range  0 mV to 320 mV

Description  This command sets the amplitude of the common mode interference in volts. Acceptable units include mV, V and exponents (for example, 100E-3 is the same as 100 mV).
This SCPI is applicable for M8041A and M8051A.

Example  :INT:CMOD:AMPL 'M2.DataOut1', 100mV

[:SOURce]:INterference[:LEVel]:CMODE:SOURce[?

Syntax  [:SOURce]:INterference[:LEVel]:CMODE:SOURce 'identifier', <HFRequency|LFRequency>
[:SOURce]:INterference[:LEVel]:CMODE:SOURce? 'identifier'

Input  'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'

Parameters  <HFRequency|LFRequency>: Select HF or LF generator.

Return Range  HFR|LFR

Description  This command selects the high frequency or low frequency generator type for common mode interference.
This SCPI is applicable for M8041A and M8051A.

Example  :INT:CMOD:SOUR 'M2.DataOut1', HFR
[:SOURce]:INTerference[:LEVEL]:CMODE:GAIN[?]

**Syntax**

- [:SOURce]:INTerference[:LEVEL]:CMODE:GAIN 'identifier', <NRf>
- [:SOURce]:INTerference[:LEVEL]:CMODE:GAIN? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'
- <NRf>: Set common mode interference gain.

**Return Range**

1.0 to 1.0

**Description**

This command sets the linear gain of the common mode interference. This command is only for M8061A and M8062A.

**Example**

```
:INT:CMOD:GAIN 'M2.DataOut1', 0.5
```

[:SOURce]:INTerference[:LEVEL]:DMODE[:STATE][?]

**Syntax**

- [:SOURce]:INTerference[:LEVEL]:DMODE[:STATE] 'identifier', <ON|OFF>
- [:SOURce]:INTerference[:LEVEL]:DMODE[:STATE]? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DATAOUT', 'M*.DATAOUT1', or 'M*.DataOut2'
- <ON|OFF>: Turn differential mode interference on/off.

**Return Range**

ON|OFF

**Description**

This command turns differential mode interference on/off. This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

**Example**

```
:INT:DMOD 'M2.DataOut1', on
```

[:SOURce]:INTerference[:LEVEL]:DMODE:AMPLitude[?]

**Syntax**

- [:SOURce]:INTerference[:LEVEL]:DMODE:AMPLitude 'identifier', <NRf>
- [:SOURce]:INTerference[:LEVEL]:DMODE:AMPLitude? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'
- <NRf>: Set differential mode interference amplitude.

**Return Range**

0 mV to 360 mV
Description  This command sets the amplitude of the differential mode interference in volts. Acceptable units include mV, V and and exponents (for example, 100E-3 is the same as 100 mV).

This SCPI is applicable for M8041A and M8051A.

Example  :INT:DMOD:AMPL ‘M2.DataOut1’, 100mV

[:SOURce]:INterference[:LEVel]:DMODE:SOURce[?]

Syntax  [:SOURce]:INterference[:LEVel]:DMODE:SOURce 'identifier', <HFRequency|LFRequency>

[:SOURce]:INterference[:LEVel]:DMODE:SOURce? 'identifier'

Input  'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'

Parameters  <HFRequency|LFRequency>: Select HF or LF generator.

Return Range  HFR|LFR

Description  This command selects the high frequency or low frequency generator type for differential mode interference.

This SCPI is applicable for M8041A and M8051A.

Example  :INT:DMOD:SOUR 'M2.DataOut1', LFR

[:SOURce]:INterference[:LEVel]:DMODE:GAIN[?]

Syntax  [:SOURce]:INterference[:LEVel]:DMODE:GAIN 'identifier', <NRf>

[:SOURce]:INterference[:LEVel]:DMODE:GAIN? 'identifier'

Input  'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'

Parameters  <NRf>: Set differential mode interference gain.

Return Range  0.01 to 1.0

Description  This command sets the linear gain of the differential mode interference.

This SCPI is applicable for M8061A and M8062A.

Example  :INT:DMOD:GAIN 'M2.DataOut1', 0.5
[:SOURce]:INTerference:ISYMbol:[STATE]?

Syntax [:SOURce]:INTerference:ISYMbol:[STATE] 'Identifier', <0|1|ON|OFF>
[:SOURce]:INTerference:ISYMbol:[STATE]? 'Identifier'

Input 'Identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters Return Range 0|1

Description This command enables/disables the inter symbol interference state.
This SCPI is applicable for M8041A, M8051A and M8062A.

Example :SOUR:INT:ISYM 'M2.DataOut1', ON

[:SOURce]:INTerference:ISYMbol:MODE?

Syntax [:SOURce]:INTerference:ISYMbol:MODE 'Identifier', <PONE | PTWO>
[:SOURce]:INTerference:ISYMbol:MODE? 'Identifier'

Input 'Identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters

Description This command allows the user to adjust inter symbol interface trace via
one (no offset) or two points (variable offset).

Inter symbol interference mode must be set to one or two points; then
only, the value of frequency and insertion loss parameter can be changed.
This SCPI is applicable for M8041A, M8051A and M8062A.

Example :SOUR:INT:ISYM:MODE 'M2.DataOut1', PTWO

[:SOURce]:INTerference:ISYMbol:FREQuency{1:2}?

Syntax [:SOURce]:INTerference:ISYMbol:FREQuency1 'Identifier', <NRf>
[:SOURce]:INTerference:ISYMbol:FREQuency2 'Identifier', <NRf>
[:SOURce]:INTerference:ISYMbol:FREQuency1? 'Identifier'
[:SOURce]:INTerference:ISYMbol:FREQuency2? 'Identifier'

Input Identifier: 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters

Description This command sets one or two discrete user definable frequencies f1 and
f2 of the inter symbol interference (ISI) trace. The mode has to be changed
to two points in order to use the second frequency.

This SCPI is applicable for M8041A, M8051A and M8062A.

Range  
1GHz to 16GHz.

Example  
:SOUR:INT:ISYM:FREQ1 'M2.DataOut1', 8.0e9

[:SOURce]:INTerference:ISYMbol:ILOSs1:2

Syntax  
[:SOURce]:INTerference:ISYMbol:ILOSs1 'Identifier', <NRf>
[:SOURce]:INTerference:ISYMbol:ILOSs2 'Identifier', <NRf>
[:SOURce]:INTerference:ISYMbol:ILOSs1? 'Identifier'
[:SOURce]:INTerference:ISYMbol:ILOSs2? 'Identifier'

Input Parameters
Identifier: 'M*.DataOut', 'M*.DataOut1', or 'M*.DataOut2'

Description  
This command sets the insertion loss at one or two discrete user definable frequencies, f1 and f2 of the inter symbol interference (ISI) trace.

This SCPI is applicable for M8041A, M8051A, or M8062A.

Range  
0.0dB to -25.0dB.

Example  

[:SOURce]:INTerference:ISYMbol:PRESet?

Syntax  
[:SOURce]:INTerference:ISYMbol:PRESet? 'Identifier'

Input Parameters
'Identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Description  
Selects an inter symbol interference (ISI) preset for the specific application.

This command should ideally not be sent in one PM (SCPI - Programm Message) with following commands...:MODE; ...:FREQuency; ...:ILOSs.

This SCPI is applicable for M8041A, M8051A and M8062A.

For M8062A the presets are available up to P11 only; please refer to the Table 48 on page -202.
The following table describes the list of presets for selected standards and traces.

Table 47  Preset List for Selected Standards and Traces (For M8041A and M8051A)

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th></th>
<th></th>
<th>P2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SCPI Name</td>
<td>f1/GHz</td>
<td>L1/dB</td>
<td>f2/GHz</td>
<td>L2/dB</td>
<td></td>
</tr>
<tr>
<td>PCIe3</td>
<td>Short</td>
<td>P1</td>
<td>1</td>
<td>-3.0</td>
<td>4</td>
<td>-9.6</td>
</tr>
<tr>
<td>PCIe3</td>
<td>Long</td>
<td>P2</td>
<td>1</td>
<td>-5.6</td>
<td>4</td>
<td>-17.9</td>
</tr>
<tr>
<td>M-PHY G3A</td>
<td>Ch1</td>
<td>P3</td>
<td>0</td>
<td>0.0</td>
<td>4.992</td>
<td>-6.0</td>
</tr>
<tr>
<td>M-PHY G3A</td>
<td>Ch2</td>
<td>P4</td>
<td>1.248</td>
<td>-4.1</td>
<td>4.992</td>
<td>-13.9</td>
</tr>
<tr>
<td>M-PHY G3B</td>
<td>Ch1</td>
<td>P5</td>
<td>0</td>
<td>0.0</td>
<td>5.83</td>
<td>-5.8</td>
</tr>
<tr>
<td>M-PHY G3B</td>
<td>Ch2</td>
<td>P6</td>
<td>1.458</td>
<td>-4.2</td>
<td>5.83</td>
<td>-13.8</td>
</tr>
<tr>
<td>M8048A</td>
<td>7.7&quot;</td>
<td>P7</td>
<td>0</td>
<td>0.0</td>
<td>4</td>
<td>-4.4</td>
</tr>
<tr>
<td>M8048A</td>
<td>9.4&quot;</td>
<td>P8</td>
<td>0</td>
<td>0.0</td>
<td>4</td>
<td>-6.0</td>
</tr>
<tr>
<td>M8048A</td>
<td>11.1&quot;</td>
<td>P9</td>
<td>1</td>
<td>-1.9</td>
<td>4</td>
<td>-6.7</td>
</tr>
<tr>
<td>M8048A</td>
<td>12.8&quot;</td>
<td>P10</td>
<td>1</td>
<td>-2.3</td>
<td>4</td>
<td>-7.1</td>
</tr>
<tr>
<td>M8048A</td>
<td>14.4&quot;</td>
<td>P11</td>
<td>1</td>
<td>-2.7</td>
<td>4</td>
<td>-8.1</td>
</tr>
<tr>
<td>M8048A</td>
<td>16.1&quot;</td>
<td>P12</td>
<td>1</td>
<td>-3.0</td>
<td>4</td>
<td>-9.0</td>
</tr>
<tr>
<td>M8048A</td>
<td>24.4</td>
<td>P13</td>
<td>1</td>
<td>-4.5</td>
<td>4</td>
<td>-13.8</td>
</tr>
<tr>
<td>SAS-3 12 Gb/s</td>
<td>P14</td>
<td>1</td>
<td>-4.5</td>
<td>4</td>
<td>-13.8</td>
<td></td>
</tr>
<tr>
<td>MIPI-Short</td>
<td>P15</td>
<td>0</td>
<td>0.0</td>
<td>5</td>
<td>-4.85</td>
<td></td>
</tr>
<tr>
<td>MIPI-Standard</td>
<td>P16</td>
<td>1.25</td>
<td>-3.75</td>
<td>5</td>
<td>-11.8</td>
<td></td>
</tr>
<tr>
<td>MIPI-Long</td>
<td>P17</td>
<td>1.25</td>
<td>-6.3</td>
<td>5</td>
<td>-20.0</td>
<td></td>
</tr>
</tbody>
</table>

Example: SOUR:INT:ISYM:PRES 'M2.DataOut1',P1
<table>
<thead>
<tr>
<th>SCPI Name</th>
<th>P1</th>
<th>f1/GHz</th>
<th>L1/dB</th>
<th>P2</th>
<th>f2/GHz</th>
<th>L2/dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEI-28G-VSR</td>
<td>Min</td>
<td>P1</td>
<td>14</td>
<td>-3.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>P2</td>
<td>14</td>
<td>-9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CEI-28G-VSR</td>
<td>100GbE</td>
<td>P3</td>
<td>12.89</td>
<td>-10.25</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CEI-28G-VSR</td>
<td>32GFC</td>
<td>P4</td>
<td>14.025</td>
<td>-10.25</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>M8048A</td>
<td>7.7”</td>
<td>P5</td>
<td>4</td>
<td>-4.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>M8048A</td>
<td>9.4”</td>
<td>P6</td>
<td>4</td>
<td>-6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>M8048A</td>
<td>11.1”</td>
<td>P7</td>
<td>1</td>
<td>-1.9</td>
<td>4</td>
<td>-6.7</td>
</tr>
<tr>
<td>M8048A</td>
<td>12.8”</td>
<td>P8</td>
<td>1</td>
<td>2.3</td>
<td>4</td>
<td>-7.1</td>
</tr>
<tr>
<td>M8048A</td>
<td>14.4”</td>
<td>P9</td>
<td>1</td>
<td>-2.7</td>
<td>4</td>
<td>-8.1</td>
</tr>
<tr>
<td>M8048A</td>
<td>16.1”</td>
<td>P10</td>
<td>1</td>
<td>-3</td>
<td>4</td>
<td>-9</td>
</tr>
<tr>
<td>M8048A</td>
<td>24.4”</td>
<td>P11</td>
<td>1</td>
<td>-4.5</td>
<td>4</td>
<td>-13.8</td>
</tr>
</tbody>
</table>
[:SOURce]:INTerference:RANDom Subnode

This subnode has the following SCPI structure:

```
[:SOURce]
  :INTerference
    :RANDom
      :HFRequency[?]
      :LFRequency[?]
      :CREStfactor[?]
      :UNIT[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:SOURce]:INTerference:RANDom:HFRequency[?]</td>
<td>[:SOURce]:INTerference:RANDom:HFRequency[?] on page 204</td>
</tr>
<tr>
<td>[:SOURce]:INTerference:RANDom:LFRequency[?]</td>
<td>[:SOURce]:INTerference:RANDom:LFRequency[?] on page 204</td>
</tr>
<tr>
<td>[:SOURce]:INTerference:RANDom:CREStfactor[?]</td>
<td>[:SOURce]:INTerference:RANDom:CREStfactor[?] on page 204</td>
</tr>
<tr>
<td>[:SOURce]:INTerference:RANDom:CREStfactor:UNIT[?]</td>
<td>[:SOURce]:INTerference:RANDom:CREStfactor:UNIT[?] on page 205</td>
</tr>
</tbody>
</table>
[:SOURce]:INTerference:RANDom:HFRequency[?]  
Syntax [:SOURce]:INTerference:RANDom:HFRequency 'identifier', <NRf>  
[:SOURce]:INTerference:RANDom:HFRequency? 'identifier'  
Input 'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'  
Parameters <NRf>: Set random interference high frequency  
Description This command sets the random interference high frequency in Hz.  
This SCPI is applicable for M8195A and M8196A.  
Example :INT:RAND:HFR 'M2.DataOut1', 1.0e9

[:SOURce]:INTerference:RANDom:LFRequency[?]  
Syntax [:SOURce]:INTerference:RANDom:LFRequency 'identifier', <NRf>  
[:SOURce]:INTerference:RANDom:LFRequency? 'identifier'  
Input 'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'  
Parameters <NRf>: Set random interference low frequency  
Description This command sets the random interference low frequency in Hz.  
This SCPI is applicable for M8195A and M8196A.  
Example :INT:RAND:LFR 'M2.DataOut1', 1.0e9

[:SOURce]:INTerference:RANDom:CREStfactor[?]  
Syntax [:SOURce]:INTerference:RANDom:CREStfactor 'identifier', <NRf>  
[:SOURce]:INTerference:RANDom:CREStfactor? 'identifier'  
Input 'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'  
Parameters <NRf>: Set random interference crest factor value  
Description This command sets the random interference crest factor value in dB or as voltage ratio depending on the currently selected crest factor unit type.  
This SCPI is applicable for M8195A and M8196A.  
Example :INT:RAND:CRES 'M2.DataOut1', 12
[:SOURce]:INTerference:RANDom:CREStfactor:UNIT[?]

Syntax

[:SOURce]:INTerference:RANDom:CREStfactor:UNIT 'identifier',
< LINear | LOGarithmic >

[:SOURce]:INTerference:RANDom:CREStfactor:UNIT? 'identifier'

Input

'identifier': 'M*.DATAOUT1' or 'M*.DataOut2'

Parameters

< LINear | LOGarithmic >

Description

This command sets the random interference crest factor unit type. Following unit types are allowed:

- LINear - This is the default setting. Select this unit to specify the crest factor as voltage ratio.
- LOGarithmic - Select this unit to specify the crest factor in dB.

This SCPI is applicable for M8195A and M8196A.

Example

:INT:RAND:CRES:UNIT 'M2.DataOut1', LOG
[:SOURce]:INTerference:SINUsoidal Subnode

This subnode controls the configuration when AWG is used as sinusoidal interference source. It has the following SCPI structure:

```
[:SOURce]
  :INTerference
    :SINusoidal
      :FREQuency[?]
      :AMPLitude{1:2}[?]
      :PHASE{1:2}[?]
      :TFRequency{1:2}[?]
      :TONemode{1:2}[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:INTerference:SINusoidal:FREQuency[?]</td>
<td>[:SOURce]:INTerference:SINusoidal:FREQuency[?] on page 207</td>
</tr>
<tr>
<td>:INTerference:SINusoidal:AMPLitude{1:2}[?]</td>
<td>[:SOURce]:INTerference:SINusoidal:AMPLitude{1:2}[?] on page 207</td>
</tr>
<tr>
<td>:INTerference:SINusoidal:PHASE{1:2}[?]</td>
<td>[:SOURce]:INTerference:SINusoidal:PHASE{1:2}[?] on page 207</td>
</tr>
<tr>
<td>:INTerference:SINusoidal:TFRequency{1:2}[?]</td>
<td>[:SOURce]:INTerference:SINusoidal:TFRequency{1:2}[?] on page 208</td>
</tr>
<tr>
<td>:INTerference:SINusoidal:TONemode{1:2}[?]</td>
<td>[:SOURce]:INTerference:SINusoidal:TONemode{1:2}[?] on page 208</td>
</tr>
</tbody>
</table>
[:SOURce]:INTerference:SINusoidal:FREQuency[?]

Syntax

[:SOURce]:INTerference:SINusoidal:FREQuency 'identifier', <NRf>
[:SOURce]:INTerference:SINusoidal:FREQuency? 'identifier'

Input

'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Parameters

This command sets the sinusoidal interference frequency in Hz.

Description

This command sets the sinusoidal interference frequency in Hz. This SCPI is applicable for M8195A and M8196A.

Example

:SOUR:INT:SIN:FREQ 'M2.DataOut1',1e9

[:SOURce]:INTerference:SINusoidal:AMPLitude{1:2}[?]

Syntax

[:SOURce]:INTerference:SINusoidal:AMPLitude{1:2} 'identifier', <NRf>
[:SOURce]:INTerference:SINusoidal:AMPLitude{1:2}? 'identifier'

Input

'identifier': 'M*.DataOut1' or 'M*.DataOut2' or 'M*.DataOut3' or 'M*.DataOut4'

Parameters

This command can be used to set the Multitone Amplitude 1 and Multitone Amplitude 2 parameters value.

Description

This command can be used to set the Multitone Amplitude 1 and Multitone Amplitude 2 parameters value. This SCPI is applicable for M8195A and M8196A.

Example

The following example shows how to use this command to set the Multitone Amplitude 1 parameter value:

:SOUR:INT:SIN:AMPL1 'M1.DataOut1',0.2

0.2

[:SOURce]:INTerference:SINusoidal:PHASe{1:2}[?]

Syntax

[:SOURce]:INTerference:SINusoidal:PHASe{1:2} 'identifier', <NRf>
[:SOURce]:INTerference:SINusoidal:PHASe{1:2}? 'identifier'

Input

'identifier': 'M*.DataOut1' or 'M*.DataOut2' or 'M*.DataOut3' or 'M*.DataOut4'

Parameters

This command can be used to set the Multitone Phase 1 and Phase 2 parameters value.

Description

This command can be used to set the Multitone Phase 1 and Phase 2 parameters value.
This SCPI is applicable for M8195A and M8196A.

Example

The following example shows how to use this command to set the Multitone Phase 1 parameter value:

:SOUR:INT:SIN:PHAS1 'M1.DataOut1',180
180

[:SOURce]:INTerference:SINusoidal:TFRequency{1:2}[?]

Syntax

[:SOURce]:INTerference:SINusoidal:TFRequency{1:2} 'identifier', <NRf>
[:SOURce]:INTerference:SINusoidal:TFRequency{1:2}? 'identifier'

Input

'identifier': 'M*.DataOut1' or 'M*.DataOut2' or 'M*.DataOut3' or 'M*.DataOut4'

Parameters

'TFRequency{1:2}': OFF | COMMon | DIFFerential

Description

This command can be used to set the Multitone Frequency 1 and Multitone Frequency 2 parameters value.

This SCPI is applicable for M8195A and M8196A.

Example

The following example shows how to use this command to set the Multitone Frequency 1 parameter value:

:SOUR:INT:SIN:TFR1 'M1.DataOut1',100000000
100000000

[:SOURce]:INTerference:SINusoidal:TONemode{1:2}[?]

Syntax

[:SOURce]:INTerference:SINusoidal:TONemode{1:2} 'identifier', OFF | COMMon | DIFFerential
[:SOURce]:INTerference:SINusoidal:TONemode{1:2}? 'identifier'

Input

'identifier': 'M*.DataOut1' or 'M*.DataOut2' or 'M*.DataOut3' or 'M*.DataOut4'

Parameters

'TONemode{1:2}': OFF | COMMon | DIFFerential

Description

This command can be used to set the Tone Mode 1 and Tone Mode 2 to either Common mode or Differential mode.

This SCPI is applicable for M8195A and M8196A.
Example  

The following example shows how to use this command to set the Tone Mode 1 to Differential Mode:

:SOUR:INT:ST1 'M1.DataOut1',DIFferential

:SOUR:INT:ST1? 'M1.DataOut1'

DIFferential
[:SOURce]:CONFigure Subnode

This subnode has the following SCPI structure:

```
[:SOURce]
  :CONFigure
  :MINTegration[?]
```

This subnode has the following command:

```
[:SOURce]:CONFigure:MINTegration[?]
```

**Syntax**

```
[:SOURce]:CONFigure:MINTegration 'identifier',
  <NONE|MUX|DMUX|BOTH>[<data-location>[<visa-resource>]]
```

```
[:SOURce]:CONFigure:MINTegration? 'identifier'
```

**Input Parameters**

- `'identifier'`: 'M*.MuxMode'
- `<NONE|MUX|DMUX|BOTH>`: Select mux/demux module.
- `<data-location>`: (Optional) Specify location of data module. If omitted or an empty string is specified, the first suitable module is chosen automatically.
- `<visa-resource>`: (Optional) Specify the address string of the N4877A.

**Return Range**

- `<NONE|MUX|DMUX|BOTH`

**Description**

This command controls the integration of a multiplexer and/or demultiplexer to a data module. Also, it controls the integration of the N4877A Clock Data Recovery and Demultiplexer.

In M8062A, the parameter values 'MUX', 'DMUX' and 'BOTH' all perform the same action i.e. they enable the 32G mode. 'NONE' disables 32G mode.

This SCPI is applicable for M8061A and M8062A.

**Example**

```
:CONF:MINT 'M2.MuxMode', BOTH
```
OUTPut Subsystem

The OUTPut subsystem controls the output ports of the pattern generator. This subsystem has the following SCPI structure:

```
:OUTPut
  ...:GLOBal
  ...
  ...:[STATe]?
  ...
  ...:[STATe]?
  ...
  ...:CONFig
  ...
  ...:SOURce[?]
  ...
  ...:DRATe?
  ...
  ...:POLarity[?]
  ...
  ...:TVOLtage[?]
  ...
  ...:TCONfig[?]
  ...
  ...:COUPling[?]
  ...
  ...:DIVider[?]
  ...
  ...:FREQuency
  ...
  ...:DIVider[?]
  ...
  ...:CALibration
  ...
  ...:SKEW[?]
  ...
  ...:EIDLe
  ...
  ...:[STATe]?
  ...
  ...:DATA
  ...
  ...:XOVer?[?]
  ...
  ...:F2Jitter[?]
  ...
  ...:[STATe]?
  ...
  ...:DEEMphasis
  ...
  ...
  ...:EINSertion
  ...
  ...
  ...:TRIGger
  ...
  ...
  ...:DEMBedding
  ...
  ...
  ...:EMBedding
  ...
  ...
  ...:ADJustment
  ...
```
This subsystem has the following commands and subnodes:

### Table 51

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:GLOBal[:STATe]?</td>
<td>:OUTPut:GLOBal[:STATe]? [on page 213]</td>
</tr>
<tr>
<td>[:STATe]?</td>
<td>:OUTPut [:STATe]? [on page 213]</td>
</tr>
<tr>
<td>:CONFig:SOURce[?]?</td>
<td>:OUTPut:CONFig:SOURce[?][on page 213]</td>
</tr>
<tr>
<td>:DRAt?</td>
<td>:OUTPut:DRAt [on page 214]</td>
</tr>
<tr>
<td>:POLarity[?]?</td>
<td>:OUTPut:POLarity[?][on page 214]</td>
</tr>
<tr>
<td>:TVOLtage[?]?</td>
<td>:OUTPut:TVOLtage[?][on page 214]</td>
</tr>
<tr>
<td>:TCONfig[?]</td>
<td>:OUTPut:TCONfig[?][on page 215]</td>
</tr>
<tr>
<td>:COUPling[?]?</td>
<td>:OUTPut:COUPling[?][on page 215]</td>
</tr>
<tr>
<td>:DIVider[?]</td>
<td>:OUTPut:DIVider[?][on page 216]</td>
</tr>
<tr>
<td>:FREQuency[?]?</td>
<td>:OUTPut:FREQuency[?][on page 216]</td>
</tr>
<tr>
<td>:EIDLe[:STATe]?</td>
<td>:OUTPut:EIDLe[:STATe]? [on page 217]</td>
</tr>
<tr>
<td>:DATA:XOVer[?]?</td>
<td>:OUTPut:DATA:XOVer[?][on page 218]</td>
</tr>
</tbody>
</table>

**Subnodes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DEEMphasis</td>
<td>:OUTPut:DEEMphasis Subnode [on page 220]</td>
</tr>
<tr>
<td>:EINSertion</td>
<td>:OUTPut:EINSertion Subnode [on page 233]</td>
</tr>
<tr>
<td>:TRIGger</td>
<td>:OUTPut:TRIGger Subnode [on page 237]</td>
</tr>
<tr>
<td>:DEMBedding</td>
<td>:OUTPut:DEMBedding Subnode [on page 239]</td>
</tr>
<tr>
<td>:EMBedding</td>
<td>:OUTPut:EMBedding Subnode [on page 242]</td>
</tr>
<tr>
<td>:ADJustment</td>
<td>:OUTPut:ADJustment Subnode [on page 245]</td>
</tr>
</tbody>
</table>
:OUTPut:GLOBal[:STATE]??

Syntax :OUTPut:GLOBal[:STATE] 'identifier', <ON|OFF|1|0>

Input 'identifier': 'M1.System'

Parameters <ON|OFF|1|0>: Enable/disable global state.

Return Range 1|0

Description This command sets global output state to ON or OFF. This command works for all modules, as it is a Global command.

Example :OUTP:GLOB 'M1.System', off

:OUTPut[:STATE]??

Syntax :OUTPut[:STATE] 'identifier', <ON|OFF|1|0>


Parameters <ON|OFF|1|0>: Switch an output on or off.

Return Range 1|0

Description This command switches an output on or off. This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8195A and M8045A.

Example :OUTP 'M1.DataOut1', off

:OUTPut:CONFig:SOURce[?]

Syntax :OUTPut:CONFig:SOURce <Identifier>, <CLEanclock|DATaclock>

Input 'identifier': 'M1.ClkOut1' or 'M1.ClkOut2'

Parameters <CLEanclock|DATaclock>

Return Range CLE|DAT
Description  This command selects the signal source for the instrument's output. This is only valid for certain clock outputs.
This query returns the present state.
This SCPI is supported by M8045A.
Example  :OUTP:CONF:SOUR 'M1.ClkOut1', CLE

:OUTPut:DRATe?
Syntax  :OUTPut:DRATe? 'identifier'
Input  'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut', or 'M1.ClkOut'
Parameters  
Description  This query returns data rate for the output channel.  
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M80195A and M8045A.
Example  :OUTP:DRAT? 'M1.DataOut1'

:OUTPut:POLarity[?] 
Syntax  :OUTPut:POLarity 'identifier', <NORMal|INVerted> 
:OUTPut:POLarity? 'identifier' 
Parameters  
<NORMal|INVerted>: Set output polarity to normal or inverted. 
Return Range  NORM|INV 
Description  This command sets the output polarity to either normal or inverted. 
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.
Example  :OUTP:POL 'M1.DataOut1', INV

:OUTPut:TVOLtage[?]
Syntax  :OUTPut:TVOLtage 'identifier', <NRf> 
:OUTPut:TVOLtage? 'identifier' 

Input Parameters

Return Range <NRf>

Description This command specifies the external termination voltage. The default is 0 V (ground). Acceptable units include mV, V and exponents (for example, 700E-3 is the same as 700 mV).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8195A.

Example :OUTP:TVOL ‘M1.DataOut1’, 700mV


Return Range BAL|UNB

Description This command selects the termination model for an output frontend:

BALanced - balanced or differential output.
UNBalanced - unbalanced with respect to ground (termination voltage).

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

Example :OUTP:TCON ‘M2.DataOut2’, BAL


Return Range AC|DC

Description <AC|DC>: Select output coupling.
Description  This command selects DC or AC output coupling.

   This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example  :OUTP:COUP 'M1.DataOut1', DC

:OUTP:DIVider[?]

Syntax  :OUTP:DIVider 'identifier', <DIV1...DIV640>

   :OUTP:DIVider? 'identifier'

Input Parameters


Specify a divider: DIV1 | DIV2 | DIV4 | DIV8 | DIV10 | DIV16 | DIV20 | DIV24 | DIV30 | DIV32 | DIV40 | DIV48 | DIV50 | DIV60 | DIV64 | DIV66 | DIV80 | DIV96 | DIV100 | DIV120 | DIV128 | DIV132 | DIV160 | DIV192 | DIV200 | DIV240 | DIV256 | DIV264 | DIV320 | DIV400 | DIV512 | DIV528 | DIV640

Return Range  DIV1 to DIV640

Description  This command selects a specific clock divider from DIV1 through DIV640.

Please note that the divider availability depends on identifier, product number and currently used symbol rate.

   This SCPI is applicable for M8041A and M8045A.

Example  :OUTP:DIV 'M1.ClkOut',DIV1

:OUTP:FREQuency[?]

Syntax  :OUTP:FREQuency 'identifier', <NRf>

   :OUTP:FREQuency? 'identifier'

Input Parameters

   'identifier': 'M1.RefClkOut' or 'M2.ClnClkOut'

Return Range  1e+7 | 1e+8

Description  This command can be used to fetch or set the frequency at RefClockOut.

   This SCPI is applicable for M8041A and M8045A. For M8062A, it is only applicable as a query.

Example  :OUTP:FREQuency 'm1.refclkout', 1e+8.
:OUTPut:FREQuency:DIVider[?]

Syntax :OUTPut:FREQuency:DIVider 'identifier', <NRf>
:OUTPut:FREQuency:DIVider? 'identifier'

Input
'identifier': 'M*.ClnClkOut'

Parameters

Return Range <NRf>

Description This command divides the frequency at clean clock out (Cln Clk Out) on an M8062A module. This SCPI is applicable for M8062A.

Example :OUTPut:FREQuency:DIVider 'M2.ClnClkout', 2

:OUTPut:CALibration:SKEW[?]

Syntax :OUTPut:CALibration:SKEW 'identifier', <NRf>
:OUTPut:CALibration:SKEW? 'identifier'

Input
'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Parameters

Return Range -2ns to 2ns

Description This command is used as offset to Data Delay and Jitter Delay. This SCPI is applicable for M8041A, M8051A and M8045A.

Example :OUTPut:CALibration:SKEW 'M1.DataOut1', 0

:OUTPut:EIDLe[:STATe][?]

Syntax :OUTPut:EIDLe[:STATe] 'identifier', <OFF|IDLE|EXTernal>
:OUTPut:EIDLe[:STATe]? 'identifier'

Input
'identifier': 'M*.DataOut'

Parameters

<OFF>: Electrical idle feature is disabled.

<IDLE>: Electrical idle feature is enabled (software controlled, power saving mode).

<EXTernal>: Electrical idle input can be used to provide an electrical idle signal at the Data Output. An external generated signal, for example by a pulse generator at the electrical idle input, controls the output stream at the Data Output.
Return Range: OFF|IDLE|EXT

Description: This command enables/disables the electrical idle feature for the Data Output. The state can be OFF, IDLE or EXTernal.

This SCPI is applicable for M8061A and M8062A.

Example: :OUTP:EIDL 'M2.DataOut', EXT

:OUTPut:DATA:XOVer[?]

Syntax: :OUTPut:DATA:XOVer 'identifier', <NRf>

:OUTPut:DATA:XOVer? 'identifier'

Input Parameters:

'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

<NRf>: Specify an eye crossover.

Return Range: For M8041A and M8051A: 10% to 90%
For M8061A and M8062A: 30% to 70%

Description: This command sets the eye crossover of the transmitter signal at the DATA OUT/-DATA OUT connectors. Crossover is defined for NRZ signals.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.


:OUTPut:DATA:F2Jitter[?]

Syntax: :OUTPut:DATA:F2Jitter 'identifier', <NRf>

:OUTPut:DATA:F2Jitter? 'identifier'

Input Parameters:

'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

<NRf>: Set the f/2 jitter.

Return Range: -20 ps to +20 ps

Description: This command sets the f/2 jitter at the output in seconds. Acceptable units include ps and exponents (for example, -10e-12 is the same as -10 ps).

This query returns the present setting.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

:OUTPut:DATA:F2Jitter:STATe[

Syntax: :OUTPut:DATA:F2Jitter:STATe 'identifier', <ON|OFF|1|0>
:OUTPut:DATA:F2Jitter:STATe? 'identifier'

Input:
'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters:

Return Range: ON|OFF|1|0

Description:
This command enables/disables f/2 jitter at the output. Enabling this command, causes stop of full clock group (all pattern generator channels and CLK/TRIG).

This query returns the present setting.

This SCPI is applicable for M8045A.

The M8070B provides an eight tap de-emphasis signal. Seven of these taps can be specified: two pre cursors and five post cursors. The resulting main cursor is calculated.

The taps can be programmed remotely in two different ways. Firstly, all the tap values can be set by one command. In this case all seven tap values have to be specified at once in a comma separated list of numbers. Total 31 preset registers are available. Secondly, a preset register can be determined as the current preset and the pre-and post-cursor commands work on this selected preset register. Only absolute values are accepted in both cases.

The sign of a tap value is defined by a separate command. If, for example, the sign of pre-cursor 2 is set to negative (-1), it will change the sign of every pre-cursor 2 in the 31 preset registers.

This subnode has the following SCPI structure:
This subnode has the following commands:

Table 52

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:UNIT[?]</td>
<td>:OUTPut:DEEMphasis:UNIT[?] on page 222</td>
</tr>
<tr>
<td>:PRESet[?]</td>
<td>:OUTPut:DEEMphasis:PRESet[?] on page 222</td>
</tr>
<tr>
<td>:PRECursor(1:2)[?]</td>
<td>:OUTPut:DEEMphasis:PRECursor(*)[?] on page 224</td>
</tr>
<tr>
<td>:POSTcursor(1:5)[?]</td>
<td>:OUTPut:DEEMphasis:POSTcursor(*)[?] on page 225</td>
</tr>
<tr>
<td>:CONFigure:SIGN[?]</td>
<td>:OUTPut:DEEMphasis:CONFigure:SIGN(*)[?] on page 225</td>
</tr>
<tr>
<td>:CURSor:MAGNitude(0</td>
<td>1</td>
</tr>
</tbody>
</table>
:OUTPut:DEEMphasis:UNIT[?]

Syntax :OUTPut:DEEMphasis:UNIT 'identifier', <DB|PCT>
         :OUTPut:DEEMphasis:UNIT? 'identifier'

Input  'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'
Parameters <DB|PCT>: Select dB or percent as the unit.

Return Range DB|PCT

Description This command defines the tap value unit: dB or PCT (PerCenT). This
command should not be sent in one PM (SCPI - Program Message) with
the following de-emphasis commands:

- :PRESet
- :PRECursor
- :POSTcursor

In other words, the unit should be set before sending the values with the
above de-emphasis commands.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and
M8045A.

Example :OUTP:DEEM:UNIT 'M1.DataOut2', dB

:OUTPut:DEEMphasis:PRESet[?]

Syntax :OUTPut:DEEMphasis:PRESet 'identifier',
        <NRf>,<NRf>,<NRf>,<NRf>,<NRf>,<NRf>,<NRf>,<NRf>

Input  'identifier': 'M*.DataOut1' or 'M*.DataOut2'
Parameters <NRf>,<NRf>,<NRf>,<NRf>,<NRf>,<NRf>,<NRf>,<NRf>:

Set register index and tap values.

<Preset Register No.>: Specify preset register number.

Return Range <Pre2>,<Pre1>,<Post1>,<Post2>,<Post3>,<Post4>,
                <Post5>

Description This command is used to enter a comma separated list representing a
register index and tap values. The first number in the list specifies the
index of the preset register. There are 31 preset registers available
addressed by a register index of 0 up to 30. The values in the list specify
the register index and taps in the following order:
Supported preset values for M8041A and M8051A are 0 - 30 and for M8061A and M8062A, 0 is the only preset value.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

Example

```
:OUTP:DEEM:PRES 'M1.DataOut2', 2,1.1,2.2,4.3,6.7,5.3,6,3.3
```

:**OUTPut:DEEMphasis:PRESet:ENABle[?]**

**Syntax**

```
:OUTPut:DEEMphasis:PRESet:ENABle 'identifier', <ON | OFF | 1 | 0>
```

**Input Parameters**

- `'identifier'`: 'M*.DataOut1' or 'M*.DataOut2'
- `<ON | OFF | 1 | 0>`

**Return Range**

`<ON | OFF | 1 | 0>`

**Description**

This command enables or disables changing the deemphasis preset register number. Using this option, user can predefine separate set of cursor values and can apply it using preset register number.

This query returns the present state.

This SCPI is applicable for M8041A, M8051A and M8062A.

Example

```
:OUTP:DEEM:PRES:ENAB 'M2.DataOut2', ON
```

:**OUTPut:DEEMphasis:SELect:PRESet[?]**

**Syntax**

```
:OUTPut:DEEMphasis:SELect:PRESet 'identifier', <NRf>
```

**Input Parameters**

- `'identifier'`: 'M*.DataOut1' or 'M*.DataOut2'
- `<NRf>`: Set current preset register.

**Return Range**

0 to 30

**Description**

This command sets the current preset register. One in 31 registers can be defined as the ‘current’ preset register. The tap values of the ‘current’ preset register are shown at the output. Preset register 0 is the default
register. The :PRECursor and :POSTCursor commands change the tap values only in that preselected preset register. There are 31 preset registers available addressed by a register index of 0 up to 30.

This command should not be sent in one PM (SCPI - Program Message) with the following de-emphasis commands:
- :PRESet
- :PRECursor
- :POSTcursor

In other words, the preset register should be selected before sending the values with the above de-emphasis commands.

Supported preset values for M8041A and M8051A are 0 - 30 and for M8061A and M8062A, 0 is the only preset value.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.


:OUTPut:DEEMphasis:PRECursor(*)[?]

Syntax  :OUTPut:DEEMphasis:PRECursor(*) 'identifier', <NRf>
          :OUTPut:DEEMphasis:PRECursor(*)? 'identifier'

Input  'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Parameters  <NRf>: Set selected pre-cursor value.

Return Range  PRECursor1: -12.04 dB to 12.04 dB | 25% to 400%
               PRECursor2: -6.02 dB to 6.02 dB | 50% to 200%

Description  This command programs the pre-cursor value of the currently selected pre-cursor (pre-cursor 1 or pre-cursor 2).

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

For M8045A, the pre-cursor values cannot be changed. You can set the pre-cursor values by adjusting the coefficient values.

Example  :OUTP:DEEM:PREC2 'M1.DataOut2', 3.1
**:OUTPut:DEEmphasis:POSTcursor(*)[?]**

**Syntax:**  
:OUTPut:DEEmphasis:POSTcursor(*) 'identifier', <NRf>  
:OUTPut:DEEmphasis:POSTcursor(*)? 'identifier'

**Input Parameters**

- **'identifier':** 'M*.DataOut1' or 'M*.DataOut2'
- **<NRf>:** Set selected post-cursor value.

**Return Range**

- POSTcursor1: -20 dB to 20 dB | 10% to 1000%
- POSTcursor2, POSTcursor3: -12.04 dB to 12.04 dB | 25% to 400%
- POSTcursor4, POSTcursor5: -6.02 dB to 6.02 dB | 50% to 200%

**Description**

This command programs the post-cursor value of the currently selected post-cursor (post-cursor 1 through post-cursor 5).

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

For M8045A, the post-cursor values cannot be changed. You can set the post-cursor values by adjusting the coefficient values.

**Example**

:OUTP:DEEM:POST4 'M1.DataOut2', 2.3

---

**:OUTPut:DEEmphasis:CONFigure:SIGN(*)[?]**

**Syntax:**  
:OUTPut:DEEmphasis:CONFigure:SIGN(*) 'identifier', <enum>, <enum>, <enum>, <enum>, <enum>, <enum>, <enum>

:OUTPut:DEEmphasis:CONFigure:SIGN(*)? 'identifier'

**Input Parameters**

- **'identifier':** 'M*.DataOut1' or 'M*.DataOut2'
- **<enum>:** Specify positive/negative tap value sign for each tap.

**Return Range**

POS|NEG (returned for each tap)

**Description**

A list of "POSitive"s and "NEGative"s specify the sign of the corresponding tap.

The following list illustrates the meaning of the comma separated list:

- <Pre2>,<Pre1>,<Post1>,<Post2>,<Post3>,<-Post4>,<Post5>

The order of the values corresponds to the order in the :PRESet command argument list. So sending the following comma separated list "pos,neg,pos,neg,neg,pos" determines that the pre-cursor 2 is a POSitive value, pre-cursor 1 is a NEGative value, post cursor 1 is a...
POSitive value and so on. This command influences all presets at once. For example, it is not possible to have a post cursor 2 with different signs in other preset registers.

This command accepts 3-taps or 7-taps sign and the meaning of the first tap changes depending upon the 3-taps or 7-taps sign sent. If 3-taps sign are sent, PreCursor1, PostCursor1, PostCursor2 will be signed. If 7-taps sign are sent, PreCursor2, PreCursor1, PostCursor1, PostCursor2, PostCursor3, PostCursor4, PostCursor5 will be signed.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

Example :OUTP:DEEM:CONF:SIGN 'M1.DataOut2',pos,pos,neg,neg,pos,pos,pos

**NOTE**

The sign (+ve or -ve) for de-emphasis values (pre-cursor or post-cursor) is set using the command "OUTPut:DEEmphasis:CONFigure:SIGN(*)". However, the same can now be directly programmed by using signed values with the commands "OUTPut:DEEmphasis:PRECursor(*)", or "OUTPut:DEEmphasis:POSTcursor(*)".

This SCPI is only supported by M8041A/M8051A (Hardware Revision 1) and M8061A modules.
:OUTP:DEEMphasis:CURSor:MAGNitude(0|1|2|3|4|5|6|7)[?]

Syntax  
:OUTP:DEEMphasis:CURSor:MAGNitude(0|1|2|3|4|5|6|7)  
 'Identifier',<NRf>  
 :OUTP:DEEMphasis:CURSor:MAGNitude(0|1|2|3|4|5|6|7)? 'identifier'

Input  
Parameters  
'M*.DataOut1' or 'M*.DataOut2'

Return Range  
<NRf>

Description  
This command sets the cursor magnitude as a signed value. The suffix selects the cursor to be addressed. The number of available cursors depends on the product number.

The M8045A supports cursors 0 to 4, where the main cursor is always cursor 2.

This query returns the present setting.

This SCPI is supported by M8045A.

Example  
:OUTP:DEEM:CURS:MAGN1 'M1.DataOut1',-0.05  

:OUTPut:DEEMphasis:CURSor:MAIN:AUTO[?]

Syntax  
:OUTPut:DEEMphasis:CURSor:MAIN:AUTO 'Identifier',<ON|OFF|1|0>  
 :OUTPut:DEEMphasis:CURSor:MAIN:AUTO? 'identifier'

Input  
Parameters  
'M*.DataOut1' or 'M*.DataOut2'

Return Range  
<ON|OFF|1|0>

Description  
This command controls whether the main cursor magnitude shall be calculated automatically in order to maintain the overall peak to peak amplitude constant, or whether the main cursor magnitude can be controlled manually.

When this is set to ON, then the main cursor magnitude is calculated to fulfill the sum below.
\[ \sum_{i=1}^{n} \text{abs}(\text{CursorMagnitude}_i) = 1 \]

This query returns the present state.

This SCPI is supported by M8045A.

**Example**

```
```

**:OUTPut:DEEMphasis:CURSor:MAIN:SUFFix?**

**Syntax**

```
:OUTP:DEEMphasis:CURSor:MAIN:SUFFix? 'Identifier'
```

**Input**

'identifier': 'M*.DataOut1' or 'M*.DataOut2'

**Return Value**

2

**Description**

This query returns the main cursor's suffix.

This SCPI is supported by M8045A.

**Example**

```
```

**:OUTPut:DEEMphasis:PCIExpress**

**Description**

When using the PCIe LTSSM on a M8045A, then the de-emphasis is controlled by a preset table. The table defines the full-swing value which will be advertised to the device under test and contains one de-emphasis definition for each of the presets that result from the full-swing selection.

Each preset consists of a coefficient value for a pre-, main-, and post cursor.

The presets are addressed by the nominals pre-cursor and post-cursor value with a resolution given by the full-swing value. This leads to pre- and post-cursor values being integer values, with a value range that depends on the full-swing value and the respective other cursor’s value.

**Example**

For a full-swing value of 63, the absolute range for the pre-cursor is 0...15 and for the post cursor it is 0...21.

The resulting nominal pre- and post-cursor de-emphasis coefficients are 0/63...15/63 and 0/63...21/63.
The preset table therefore provides a means of calibrating the pre-, main-, and post-cursor for each of the valid nominal cursor settings of the PCIe LTSSM.

**Import preset definitions into the workspace**

Use the following SCPI commands to import user defined preset tables into the M8070B workspace

```scpi
:MMEMory:WORKspace:IMPort[:USER]:CATalog? PCIDeemphasis
:MMEMory:WORKspace:IMPort[:USER] PCIDeemphasis,<"SourcePath">,<"RelativeTargetPath">
:MMEMory:WORKspace:IMPort[:USER]:DELete PCIDeemphasis,<"RelativeTargetPath">
```

M8070B contains default preset tables for all supported full-swing values (24 .. 63). These can be used as a starting point for external calibration of the de-emphasis presets.

The naming scheme for these default preset-tables is as follows:

Factory/FullSwing-XX.xml

where XX is the full-swing value (24 .. 63).

The calibration code will have to find the de-emphasis coefficient for the pre-cursor, main-cursor and post-cursor that results in the pre-shoot and de-emphasis given by Pre/FullSwing and Post/FullSwing for each of the entries in the Presets list.
The content of the preset definition is defined like this:

```xml
<?xml version="1.0" encoding="utf-16"?>
<PCIeLtssmDeemphasisPresets
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Version>1.0</Version>
  <Description>De-emphasis tap gain presets for the PCIe LTSSM</Description>
  <FullSwing>24</FullSwing>
  <Presets>
    <P>
      <Pre>0</Pre>
      <Post>0</Post>
      <Coefficients>0,1,0</Coefficients>
    </P>
    <!-- entries removed for clarity -->
    <P>
      <Pre>6</Pre>
      <Post>2</Post>
      <Coefficients>-0.25,0.666666666666667,-0.0833333333333333</Coefficients>
    </P>
  </Presets>
</PCIeLtssmDeemphasisPresets>
```
:OUTPut:DEEMphasis:PCIExpress:PRECursor

Syntax :OUTPut:DEEMphasis:PCIExpress:PRECursor 'Identifier',<NRf>

Input Parameters

'dentifier': 'M*.DataOut1' or 'M*.DataOut2'

Description This command sets the pre-cursor to the given value, with a nominal cursor coefficient of pre-cursor/full-scale, e.g. 7/63 for a full-scale value of 63.

The PCIe de-emphasis presets are only available when the pattern sequence is configured for the PHY protocol PCIe3, PCIe4 or PCIe5. When the sequence is not using the PCIe LTSSM, then the de-emphasis is controlled via the cursor magnitudes directly.

This SCPI is only supported by M8045A.

Example :OUTP:DEEM:PCIE:PRE 'M1.DataOut1',7

:OUTPut:DEEMphasis:PCIExpress:POSTcursor

Syntax :OUTPut:DEEMphasis:PCIExpress:POSTcursor 'Identifier',<NRf>

Input Parameters

'dentifier': 'M*.DataOut1' or 'M*.DataOut2'

Description This command sets the post-cursor to the given value, with a nominal cursor coefficient of post-cursor/full-swing, e.g. 11/63 for a full-swing value of 63.

The PCIe de-emphasis presets are only available when the pattern sequence is configured for the PHY protocol PCIe3, PCIe4 or PCIe5. When the sequence is not using the PCIe LTSSM, then the de-emphasis is controlled via the cursor magnitudes directly.

This SCPI is only supported by M8045A.

Example :OUTP:DEEM:PCIE:POST 'M1.DataOut1',11


Input Parameters:
- 'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Description:
This command selects the de-emphasis preset definition for the PCIe LTSSM.

When switching the preset definition table, the software will try to maintain the cursor coefficients as constant as possible. The accuracy of this adoption depends on the resolution of the initial and new preset table. Due to the fractional nature of the nominal coefficients, there may be a small variation in the generated de-emphasis.

For example, when changing from a setting with full-swing 46, pre-cursor 5 and post-cursor 11 to a preset table with a full-swing value of 63, the pre- and post-cursor setting will be adapted. But the resulting signal will still vary a bit, because 5/46 cannot be expressed as x/63 with x being an integral value.

The PCIe de-emphasis presets are only available when the pattern sequence is configured for the PHY protocol PCIe3, PCIe4 or PCIe5. When the sequence is not using the PCIe LTSSM, then the de-emphasis is controlled via the cursor magnitudes directly, but the preset- table for the LTSSM can still be selected.

This SCPI is only supported by M8045A.

:OUTPut:EINSertion Subnode

This subnode has the following SCPI structure:

```
:OUTPut
- EINSertion
  - [:STATE][?]
  - [:RATio][?]
  - [:MODE][?]
  - [:ONCE]
  - [:DISParity]
```

This subnode has the following commands:

**Table 53**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:STATE][?]</td>
<td>:OUTPut:EINSertion[:STATE][?] on page 234</td>
</tr>
<tr>
<td>[:RATio][?]</td>
<td>:OUTPut:EINSertion[:RATio][?] on page 234</td>
</tr>
<tr>
<td>[:MODE][?]</td>
<td>:OUTPut:EINSertion[:MODE][?] on page 235</td>
</tr>
</tbody>
</table>
**:OUTP:EINS[ERT]:[:STATe][:?]**

Syntax :OUTP:EINS[ERT]:[:STATe] 'identifier', <ON|OFF|1|0>

:OUTP:EINS[ERT]:[:STATe]? 'identifier'

Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters <ON|OFF|1|0>: Switch error insertion output on or off.

Description This command switches the error insertion feature on or off for the specified output.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example :OUTP:EINS 'M1.DataOut1', on

**:OUTP:EINS[ERT]:RATio[:?]**

Syntax :OUTP:EINS[ERT]:RATio 'identifier', <RM1|...|RM12>

:OUTP:EINS[ERT]:RATio? 'identifier'

Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters <RM1|...|RM12>: Set the error insertion ratio.

Return Range <RM1|...|RM12>

Description This command controls the ratio of the internal fixed error insertion ratio. The range is 1E-12 (RM12) to 1E-1 (RM1).

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example :OUTP:EINS:RAT 'M1.DataOut1', RM6
:OUTPut:EINSertion:MODE[?]

Syntax :OUTPut:EINSertion:MODE 'identifier', <ERFSpacing|...|SINB>

Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters <ERFSpacing|...|SINB>: Specify the error insertion mode.

Return Range <ERFSpacing|...|SINB>

Description This command is used to specify the error insertion mode as follows:

ERFSpacing: Generates errors with a predetermined bit error rate (fixed spacing).

ERATio: Generates errors with a predetermined bit error rate. Erroneous bits are randomly distributed.

CINA: Specifies CTRL IN A as a trigger.

CINB: Specifies CTRL IN B as a trigger.

BRK: Specifies software break command as a trigger.

SINA: Specifies SYS IN A as a trigger source.

SINB: Specifies SYS IN B as a trigger source.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example :OUTP:EINS:MODE 'M1.DataOut1', ERFSspacing

:OUTPut:EINSertion:ONCE

Syntax :OUTPut:EINSertion:ONCE 'identifier'

Input 'identifier': 'M*.DataOut1', 'M*.DataOut2', or 'M*.DataOut'

Parameters Description This command generates a single error (single bit error) on the specified output.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8045A.

Example :OUTP:EINS:ONCE 'M1.DataOut1'
:OUTPut:EINSertion:DISParity

Syntax  :OUTPut:EINSertion:DISParity 'identifier'

Input   'identifier': 'M*.DataOut1' or 'M*.DataOut2'

Parameters

Description This command generates a single disparity error (only valid for 8b/10b operating mode) on the specified output.

This SCPI is applicable for M8041A, M8051A and M8045A.

Example  :OUTP:EINS:DISP 'M1.DataOut2'
This subnode has the following SCPI structure:

```
:OUTPut
  :TRIGger
     :MODE[?]
     :DIVider[?]
```

This subnode has the following commands:

### Table 54

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MODE[?]</td>
<td>:OUTPut:TRIGger:MODE[?] on page 237</td>
</tr>
<tr>
<td>:DIVider[?]</td>
<td>:OUTPut:TRIGger:DIVider[?] on page 238</td>
</tr>
</tbody>
</table>

#### :OUTPut:TRIGger:MODE[?]

**Syntax**: :OUTPut:TRIGger:MODE 'identifier', <CLOCK|SEQUencer>

- :OUTPut:TRIGger:MODE? 'identifier'
- :OUTPut:TRIGger:MODE 'identifier', <CLOCK|SEQUencer>

**Input Parameters**

- 'identifier': 'M1.TrigOut'

**Return Range**

- CLOC|SEQ

**Description**

This command sets the operating mode of the trigger out (TRIG OUT) either to subrate clock or sequencer controlled mode.

This SCPI is applicable for M8041A and M8045A.

**Example**

:OUTP:TRIG:MODE 'M1.TrigOut', CLOC
:**OUTPut:TRIGger:DIVider**

Syntax: `:OUTPut:TRIGger:DIVider 'identifier', <NRf>`

- **Input**
  - 'identifier': 'M1.TrigOut'
- **Parameters**
  - `<NRf>`: Set the divider factor for the trigger out.
- **Return Range**: 2 to 65535
- **Description**: This command sets the divider factor of the trigger out when the subrate clock operating mode is selected.
  
  This SCPI is applicable for M8041A and M8045A.
- **Example**: `:OUTPut:TRIG:DIV 'M1.TrigOut', 8`
:OUTPut:DEMBedding Subnode

This subnode has the following SCPI structure:

```
:OUTPut
  DEMBedding
    :SCABle
    [:STATe][?]
    SPARameter
      [:STATe][?]
      :PROFile[?]
      :IPORt[?]
      :OPORt[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SCABle:STATe[?]</td>
<td>:OUTPut:DEMBedding:SCABle:STATe[?] on page 240</td>
</tr>
<tr>
<td>SPARameter:STATe[?]</td>
<td>:OUTPut:DEMBedding:SPARameter:STATe[?] on page 240</td>
</tr>
<tr>
<td>SPARameter:IPORt[?]</td>
<td>:OUTPut:DEMBedding:SPARameter:IPORt[?] on page 241</td>
</tr>
<tr>
<td>SPARameter:OPORt[?]</td>
<td>:OUTPut:DEMBedding:SPARameter:OPORt[?] on page 241</td>
</tr>
</tbody>
</table>
\textbf{:OUTPut:DEMBedding:SCABle:STATe[?]} 

\textbf{Syntax} 
:OUTPut:DEMBedding[:SCABle][:STATe] 'identifier', \{OFF | ON | 0 | 1\}  
:OUTPut:DEMBedding[:SCABle][:STATe]? 'identifier'

\textbf{Input} 
'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', or 'M*.DataOut4'

\textbf{Parameters} 
OFF | ON | 0 | 1: Enables/disables standard cable compensation.

\textbf{Return Range} 
0 | 1

\textbf{Description} 
This command switches on/off the standard cable compensation on the specified output.

The query return the present setting.

This SCPI is applicable for M8195A and M8196A.

\textbf{Example} 
:OUTP:DEMB:SCAB:STAT 'M1.DataOut1', ON

\textbf{:OUTPut:DEMBedding:SPARameter:STATe[?]} 

\textbf{Syntax} 
:OUTPut:DEMBedding:SPARameter[:STATe] 'identifier', \{OFF | ON | 0 | 1\}  
:OUTPut:DEMBedding:SPARameter[:STATe]? 'identifier'

\textbf{Input} 
'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', or 'M*.DataOut4'

\textbf{Parameters} 
<OFF | ON | 0 | 1>

\textbf{Description} 
This command switches on/off the compensation via S-parameter file on the specified output.

The query return the present setting.

This SCPI is applicable for M8195A and M8196A. For M8045A, only S2P files are supported.

\textbf{Example} 
:OUTP:DEMB:SPAR:STAT 'M1.DataOut1', ON

\textbf{:OUTPut:DEMBedding:SPARameter:PROFile[?]} 

\textbf{Syntax} 
:OUTPut:DEMBedding:SPARameter:PROFile 'identifier', {'Filename'}  

\textbf{Input} 
'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', or 'M*.DataOut4'

\textbf{Parameters} 
This command selects the S-Parameter profile. S-parameter is defined by frequency/real/imaginary tuples for 1 to 9 port components.
The query return the present setting.
This SCPI is applicable for M8195A and M8196A.

Example:
```
'factory/huber_suhner_cable.s2p'
```

`:OUTPut:DEMBedding:SPARameter:IPORt[?]`

**Syntax**
```
:OUTPut:DEMBedding:SPARameter:IPORt 'identifier', <NRf>
:OUTPut:DEMBedding:SPARameter:IPORt? 'identifier'
```

**Input Parameters**
- `'identifier'`: 'M*.DataOut1','M*.DataOut2','M*.DataOut3',or 'M*.DataOut4'
- `1|2|3|4`

**Return Range**
1|2|3|4; depends upon the selected S parameter file.

**Description**
This command selects the input port in the S-Parameter profile.
The query return the present setting.
This SCPI is applicable for M8195A and M8196A.

Example:
```
:OUTP:DEMB:SPAR:IPOR 'M1.DataOut1', 1
```

`:OUTPut:DEMBedding:SPARameter:OPORt[?]`

**Syntax**
```
:OUTPut:DEMBedding:SPARameter:OPORt 'identifier', <NRf>
:OUTPut:DEMBedding:SPARameter:OPORt? 'identifier'
```

**Input Parameters**
- `'identifier'`: 'M*.DataOut1','M*.DataOut2','M*.DataOut3',or 'M*.DataOut4'
- `1|2|3|4`

**Return Range**
1|2|3|4; depends upon the selected S-parameter file.

**Description**
This command selects the output port in the S-Parameter profile.
The query return the present setting.
This SCPI is applicable for M8195A and M8196A.

Example:
```
```
:OUTPut:EMBedding Subnode

This subnode has the following SCPI structure:

```
:OUTPut
  :EMBedding
  :SPARameter
    [:STATe][?]
    [:PROFile][?]
    [:IPORT][?]
    [:OPORT][?]
```

This subnode has the following commands:

Table 56

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SPARameter[:STATe][?]</td>
<td>:OUTPut:EMBedding:SPARameter[:STATe][?] on page 243</td>
</tr>
<tr>
<td>:SPARameter:OPORT[?]</td>
<td>:OUTPut:EMBedding:SPARameter:OPORT[?] on page 244</td>
</tr>
</tbody>
</table>
OUTPut:EMBedding:SPARameter[:STATe][?]

Syntax :OUTPut:EMBedding:SPARameter[:STATe] 'identifier', <OFF | ON | 0 | 1>
:OUTPut:EMBedding:SPARameter[:STATe]? 'identifier'

Input 'identifier': 'M*.DataOut1','M*.DataOut2','M*.DataOut3', or 'M*.DataOut4'

Parameters OFF | ON | 0 | 1

Return Range 0|1

Description This command Switches on/off the compensation via S-parameter file on
the specified output.

The query returns the present setting.

This SCPI is applicable for M8195A and M8196A. For M8045A, only S2P
files are supported.

Example :OUTP:EMB:SPAR:STAT 'M1.DataOut1', ON

OUTPut:EMBedding:SPARameter:PROFile[?]


Input 'identifier': 'M*.DataOut1','M*.DataOut2','M*.DataOut3', or 'M*.DataOut4'

Parameters

Description This command selects the S-Parameter profile. S-parameter is defined by
frequency/real/imaginary tuples for 1 to 9 port components.

The query returns the present setting.

This SCPI is applicable for M8195A and M8196A.

Example OUTP:EMB:SPAR:PROF 'M1.DataOut1', 'factory/huber_suhner_cable.s2p'

OUTPut:EMBedding:SPARameter:IPOrt[?]

Syntax :OUTPut:EMBedding:SPARameter:IPOrt 'identifier', <NRf>

Input 'identifier': 'M*.DataOut1','M*.DataOut2','M*.DataOut3', or 'M*.DataOut4'

Parameters

Description This command selects the input port in the S-Parameter profile.

The query returns the present setting.
This SCPI is applicable for M8195A and M8196A.

Example: `:OUTP:EMB:SPAR:IPOR 'M1.DataOut1', 1`

`:OUTPut:EMBedding:SPARameter:OPORt[?]`

Syntax: `:OUTPut:EMBedding:SPARameter:OPORt 'identifier', <NRf>`

`:OUTPut:EMBedding:SPARameter:OPORt? 'identifier'`

Input Parameters:
- `'identifier': 'M*.DataOut1', 'M*.DataOut2', 'M*.DataOut3', or 'M*.DataOut4'`

Description: This command selects the output port in the S-Parameter profile.

The query returns the present setting.

This SCPI is applicable for M8195A and M8196A.

Example: `OUTP:EMB:SPAR:OPOR 'M1.DataOut1', 2`
:OUTPut:ADJJustment Subnode

This subnode has the following SCPI structure:

```
:OUTPut
  :ADJJustment
    :ABORt
    :STATus
    [:VALue]?  
    :BIND
    :DEEMphasis
    :AUTO
```

**NOTE**

M8045A is the only instrument capable of supporting these commands for DataOut locations.

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ABORt</td>
<td>:OUTPut:ADJJustment:ABORt on page 246</td>
</tr>
<tr>
<td>:STATus [:VALue]?</td>
<td>:OUTPut:ADJJustment:STATus [:VALue]? on page 246</td>
</tr>
<tr>
<td>:BIND</td>
<td>:OUTPut:ADJJustment:BIND on page 247</td>
</tr>
</tbody>
</table>
:OUTPut:ADJJustment:ABORt

Syntax :OUTPut:ADJJustment:ABORt 'identifier'

Parameters

Input 'identifier': 'M1.DataOut1','M2.DataOut2'

Description This command interrupts the adjustments.

Example :OUTP:ADJ:ABOR 'M1.DataOut1'

:OUTPut:ADJJustment:STATus [:VALue]?

Syntax :OUTPut:ADJJustment:STATus [:VALue]? 'Identifier'

Parameters Any pattern generator location and group name. Example: 'M1.DataOut1', 'M1.DataOut2'

Description This query retrieves adjustment status information per pattern generator or pattern generator group.

This query returns the following values:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPROGRESS</td>
<td>At least one addressed pattern generator is executing an adjustment procedure.</td>
</tr>
<tr>
<td>ABORTED</td>
<td>The adjustment procedure has been aborted by the user before all pattern generator channels finished the procedure.</td>
</tr>
<tr>
<td>FAILED</td>
<td>All addressed pattern generator channels have finished the adjustment procedure, but at least one was unable to finish successfully.</td>
</tr>
<tr>
<td>SUCCESS</td>
<td>The addressed pattern generator channels have successfully finished the adjustment procedure. Pattern Generator channels that are addressed by the command but return UNKNOWN state, do not contribute to the result.</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>None of the addressed pattern generators has executed an adjustment procedure yet.</td>
</tr>
</tbody>
</table>

Note This query cannot be used in the same compound SCPI statement that is starting the adjustment procedure. If this query is sent to the instrument right after the command that is initiating the adjustment procedure, then the returned value is undefined and can reflect the state that was valid before the adjustment. This can be overcome by reading any value from the instrument (e.g. *ESE?), except for *OPC?, as this will block until the adjustment procedure is finished.

Example :OUTP:ADJ:STAT? 'M1.DataOut1'
":"OUTPut:ADJustment:BIND
Syntax :OUTPut:ADJustment:BIND 'Identifier', 'measurement channel identifier'
Input Identifier: 'M1.DataOut1' or 'M1.DataOut2'
Parameters Measurement Channel Identifier:
'DCA*.Slot*.Channel 1A' or 'DCA*.Slot*.Channel 2A'
Description This command selects the location of the oscilloscope channel that shall be used by the addressed pattern generator channel.
This is required to automatically determine and set the deemphasis coefficients at the end of the cable.

:"OUTPut:ADJustment:DEEMphasis:AUTO
Syntax :OUTPut:ADJustment:DEEMphasis:AUTO 'Identifier'
Input 'M1.DataOut1' or 'M1. DataOut2'
Parameters
Description This command automatically determines and sets the deemphasis coefficients at the end of cable.
Example :OUTP:ADJ:DEEM:AUTO 'M1.DataOut1'
EVENt Subsystem

This subsystem has the following SCPI structure:

```
:EVENt
  :ADVance[?]
  :STATE[?]
  :SOURce[?]
```

This subsystem has the following commands:

**Table 58**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:EVENt:ADVance[?]</td>
<td>:EVENt:ADVance[?] on page 248</td>
</tr>
<tr>
<td>:EVENt:ADVance:STATE[?]</td>
<td>:EVENt:ADVance:STATE[?] on page 249</td>
</tr>
<tr>
<td>:EVENt:ADVance:SOURce[?]</td>
<td>:EVENt:ADVance:SOURce[?] on page 249</td>
</tr>
</tbody>
</table>

**:EVENt:ADVance[?]**

**Syntax**

:EVENT:ADVance[?] 'identifier'

:EVENT:ADVance? 'identifier'

**Input Parameters**

'identifier': 'M1.EventConfig'

**Description**

This command asserts the advance event in the M819xA Soft Front Panel. The query returns the present setting.

This SCPI is applicable for M8195A and M8196A.

**Example**

:EVENT:ADV:SOUR 'M1.EventConfig', TRIG

:EVENT:ADV:SOUR? 'M1.EventConfig'

TRIG
:EVENT:ADVance:STATe[?]

Syntax  
:EVENT:ADVance:STATe[?] 'identifier', <1 | 0 | ON | OFF>
:EVENT:ADVance:STATe? 'identifier'

Input  'identifier': 'M1.EventConfig'

Parameters  1 | 0 | ON | OFF

Description  This command enables/disables the hardware driven advance event in the Soft Front Panel. Triggering this event can also be done via software.

The query returns the present setting.

This SCPI is applicable for M8195A and M8196A.

Example  
:EVENT:ADV:STAT 'M1.EventConfig', 1
1

:EVENT:ADVance:SOURce[?]

Syntax  
:EVENT:ADVance:SOURce[?] 'identifier', <TRIGger | EVENt>
:EVENT:ADVance:SOURce? 'identifier'

Input  'identifier': 'M1.EventConfig'

Parameters  TRIGger | EVENt

Description  This command allows to select whether the Trigger In or the Event In is used as hardware input for advance event.

The query returns the present setting.

This SCPI is applicable for M8195A and M8196A.

Example  
:EVENT:ADV:SOUR 'M1.EventConfig', TRIG
:EVENT:ADV:SOUR? 'M1.EventConfig'
TRIG
INPut Subsystem

The INPut subsystem controls the input ports of the analyzer.

This subsystem has the following SCPI structure:

```
:INPut
  :-:BRM[?]
  :-:DRATe?
  :-:DELay[?]
  :-:POLarity[?]
  :-:CMVoltage[?]
    :-:DETect
  :-:THReshold[?]
  :-:TVOLtage[?]
  :-:TCONfig[?]
  :-:CMODE[?]
  :-:COUPling[?]
  :-:SENSitivity[?]
  :-[:VOLTage]
    :-:WINDow[?]
      :-:ALLocate[?]
        :-:DETect
        :- [:NRZ]:THReshold[?]
        :-:PAM4[:SYMBol]
          :-:THReshold(1|2|3)[?]
      :-:CDR
    :-...
  :-:ALIGnment
    :-...
  :-:EQUalization
    :-...
  :-:SQUElch
    :-...
```
The subsystem has the following commands and subnodes:

Table 59

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:BRM[?]</td>
<td>:INPut:DATA:BRMode[?] on page 252</td>
</tr>
<tr>
<td>:DELay</td>
<td>:INPut:DELay[?] on page 253</td>
</tr>
<tr>
<td>:POLarity</td>
<td>:INPut:POLarity[?] on page 253</td>
</tr>
<tr>
<td>:CMVoltage[?]</td>
<td>:INPut:CMVoltage[?] on page 254</td>
</tr>
<tr>
<td>:CMVoltage:DETect</td>
<td>:INPut:CMVoltage:DETect on page 254</td>
</tr>
<tr>
<td>:THReshold</td>
<td>:INPut:THReshold[?] on page 254</td>
</tr>
<tr>
<td>:TVOLtage</td>
<td>:INPut:TVOLtage[?] on page 255</td>
</tr>
<tr>
<td>:TCONfig[?]</td>
<td>:INPut:TCONfig[?] on page 255</td>
</tr>
<tr>
<td>:CMODE[?]</td>
<td>:INPut:CMODE[?] on page 256</td>
</tr>
<tr>
<td>COUPling[?]</td>
<td>:INPut:COUPling[?] on page 257</td>
</tr>
<tr>
<td>SENSitivity[?]</td>
<td>:INPut:SENSitivity[?] on page 257</td>
</tr>
<tr>
<td>[:VOLTage]:WINDow[?]</td>
<td>:INPut[:VOLTage]:WINDow[?] on page 257</td>
</tr>
<tr>
<td>[:VOLTage]:WINDow:ALLocate[?]</td>
<td>:INPut[:VOLTage]:WINDow:ALLocate[?] on page 258</td>
</tr>
<tr>
<td>[:VOLTage]:WINDow:DETect</td>
<td>:INPut[:VOLTage]:WINDow:DETect on page 259</td>
</tr>
<tr>
<td>:INPut[:VOLTage][:NRZ]:THReshold[?]</td>
<td>:INPut[:VOLTage][:NRZ]:THReshold[?] on page 259</td>
</tr>
<tr>
<td>:INPut[:VOLTage]:PAM4[:SYMBol]:THReshold(1</td>
<td>2</td>
</tr>
</tbody>
</table>

Subnodes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:CDR</td>
<td>:INPut:CDR Subnode on page 261</td>
</tr>
<tr>
<td>:ALIGNment</td>
<td>:INPut:ALIGNment Subnode on page 271</td>
</tr>
<tr>
<td>:EQUalization</td>
<td>:INPut:EQUalization Subnode on page 279</td>
</tr>
<tr>
<td>:SQUEIch</td>
<td>:INPut:SQUEIch Subnode on page 296</td>
</tr>
</tbody>
</table>
:INPut:DATA:BRMode[?]

Syntax: :INPut:DATA:BRMode 'identifier', < 0 | 1 | ON | OFF>

:INPut:DATA:BRMode? 'identifier'

Input Parameters

'identifier': 'M*.DataIn1' or 'M*.DataIn2'

Range: 0 | 1 | ON | OFF

Description: This command enables/disables the bit recovery mode. In BRM, the error detector does not expect any specific data. This mode can be used if the incoming data is completely unknown.

Example: :INP:DATA:BRM 'M1.DataIn1',ON

NOTE

The bit recovery mode is only available with CDR.

:INPut:DRATe?

Syntax: :INPut:DRATe? 'identifier'

Input Parameters

'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*.DataIn'

Description: This query returns the data rate for the input channel. This SCPI is applicable for M8061A M8062A and M8046A.

Example: :INP:DRAT? 'M1.DataIn1'
:INPut:DELay[?]

Syntax :INPut:DELay 'identifier', <NRf>

:INPut:DELay? 'identifier'

Input 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*.DataIn', 'INF*.DataIn1'

Parameters <NRf>: Specify sampling edge delay value.

Return Range For M8041A, M8051A and M8046A: -6.7 ns to +6.7 ns
For M8061A: 0 ps to 60 ps
For M8062A: -10 ns to 10 ns
For M8046A: -6.7 ns to 6.7 ns

Description This command specifies the delay of the sampling edge.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8046A and DSO634A, DSA634A, DSAX96204A and DSOX96204A.

Example :INP:DEL 'M1.DataIn1', 1e-9

:INPut:POLarity[?]

Syntax :INPut:POLarity 'identifier', <NORMal | INVerted | EITHer>

:INPut:POLarity? 'identifier'

Input 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*.DataIn', 'INF*.DataIn1'

Parameters <NORMal | INVerted | EITHer>: Set input polarity. The EITHer option only works with the M8195A AWG module.

Return Range NORM | INV | EITH

Description This command sets the input polarity of the specified input to either NORMal, INVerted or EITHer.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8046A, M8195A, DSO634A, DSA634A, DSAX96204A and DSOX96204A.

Example :INP:POL 'M1.DataIn1', INV
:INPut:CMVoltage[?]  
**Syntax**: :INPut:CMVoltage 'identifier', <NRf>  
:INPut:CMVoltage? 'identifier'  
**Input**  
'identifier': 'M*.DataIn1' or 'M*.DataIn2'  
**Parameters**  
<NRf>: Set common mode voltage.  
**Return Range**  
-1.0 V to 2.6 V; The range is dependent on other system parameters. See Command Syntax to Find Min/Max Values on page 109.  
**Description**  
This command is used to measure the common mode voltage of the selected data input by the module itself. Acceptable units are mV, V and exponents (for example, 300E-3 is the same as 300 mV).  
This SCPI is applicable for M8041A and M8051A.  
**Example**  
:INP:CMV 'M1.DataIn1', 300mV

:INPut:CMVoltage:DETect  
**Syntax**: :INPut:CMVoltage:DETect 'identifier'  
**Input**  
'identifier': 'M*.DataIn1' or 'M*.DataIn2'  
**Parameters**  
**Description**  
This command measures the common mode voltage at the selected data input and applies it to the comparator.  
This SCPI is applicable for M8041A and M8051A.  
**Example**  
:INP:CMV:DET 'M1.DataIn1'

:INPut:THReshold[?]  
**Syntax**: :INPut:THReshold 'identifier', <NRf>  
:INPut:THReshold? 'identifier'  
**Input**  
'identifier': 'M*.DataIn1', 'M*.DataIn2', 'M2.ElectIdleIn', 'M*DataIn', 'SysIn A/B' or 'CtrlIn A/B'  
**Parameters**  
<NRf>: Set receiver input threshold voltage.  
**Return Range**  
For M8041A and M8051A: -1 V to 1 V  
For M8062A: -800 mV to 800 mV  
For M8046A: -250 mV to 250 mV
Description  This command sets the threshold voltage of the selected receiver input connector (decision threshold). Acceptable units are mV, V and exponents (for example, 300E-3 is the same as 300 mV).
This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

Example  :INP:THR 'M1.DataIn1', 200 mV

:INPut:TVOLtage[?]

Syntax  :INPut:TVOLtage 'identifier', <NRf>
:INPut:TVOLtage? 'identifier'

Input Parameters  'identifier': 'M*.DataIn1', 'M*.DataIn2', or 'M2.ElectIdleIn'
<NRf>: Specify receiver input termination voltage.

Return Range  -1 V to 2.9 V; The range is dependent on other system parameters. See Command Syntax to Find Min/Max Values on page 109.

Description  This command specifies the termination voltage of the selected receiver input connector. Acceptable units are mV, V and exponents (for example, 300E-3 is the same as 300 mV).
This SCPI is applicable for M8041A, M8051A and M8062A.

Example  :INP:TVOL 'M1.DataIn1', 300mV

:INPut:TCONfig[?]

Syntax  :INPut:TCONfig 'identifier', <BALanced|UNBalanced>
:INPut:TCONfig? 'identifier'

Input Parameters  'identifier': 'M*.DataIn1' or 'M*.DataIn2'
<BALanced|UNBalanced>: Select input termination configuration.

Return Range  BAL|UNB

Description  This command selects the termination configuration for an input front end. The following options are available:
BALanced: balanced or differential
UNBalanced: unbalanced with respect to ground (termination voltage)
This SCPI is applicable for M8041A and M8051A.

Example  :INP:TCON 'M1.DataIn1', BAL
:INPut:CMODe[?]

Syntax
:INPut:CMODe 'identifier',
<SENormal|SEComplement|DIFFerential|SINGleended>
:INPut:CMODe? 'identifier'

Input Parameters
'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn', 'INF*.DataIn1'

Return Range
SEN|SEC|DIFF|SING

Description
This command defines the compare mode. This command defines which
input (DATA or DATA) is active. The following options are available:

SENNormal: Data stream is compared at Normal input; Complement is
terminated or connected to the complement output of the source. This
allows to measure a single-ended signal as well as the non-inverted signal
of a differential pair.

SEComplement: Data stream is compared at Complement input; Normal
is terminated or connected to the normal output of the source. This allows
to measure a single-ended signal as well as the inverted signal of a
differential pair.

DIFFerential: If differential both input ports need to receive a signal. The
actual data signal is measured as the voltage difference between the two
incoming signals.

SINGleended: Data stream is provided at either Normal input or
Complement input. The other input is either left open or terminated with
50 Ohm.

Note that the availability of the particular selection depends on the
currently used hardware.

This SCPI is applicable for M8041A, M8051A, M8061A, M8046A and
real-time oscilloscope based error detector channels.

Example
:INP:CMOD 'M1.DataIn1', SEC
**:INPut:COUPling[?]**

**Syntax**
:INPut:COUPling 'identifier', <AC|DC>

:INPut:COUPling? 'identifier'

**Input**
'identifier': 'M*.DataIn1' or 'M*.DataIn2'

**Parameters**
<AC|DC>: Select input coupling.

**Return Range**
AC|DC

**Description**
This command selects DC or AC input coupling. This SCPI is applicable for M8041A and M8051A.

**Example**
:INP:COUP 'M1.DataIn1', DC

**:INPut:SENSitivity[?]**

**Syntax**
:INPut:SENSitivity 'identifier', <HIGH|NORMal>

:INPut:SENSitivity? 'identifier'

**Input**
'identifier': 'M*.DataIn1' or 'M*.DataIn2'

**Parameters**
<HIGH|NORMal>: Select input coupling.

**Return Range**
HIGH|NORM

**Description**
This command sets the error detector input sensitivity to high or normal. This SCPI is applicable for M8041A, M8051A and M8062A.

**Example**
:INP:SENS 'M1.DataIn1', HIGH

**:INPut[:VOLTage]:WINDow[?]**

**Syntax**
:INPut[:VOLTage]:WINDow 'Identifier', < NRf>

:INPut[:VOLTage]:WINDow? 'Identifier'

**Input**
'identifier': 'M*.DataIn' or 'INF*.DataIn1'

**Parameters**
<NRf>: Select input coupling.

**Description**
This command defines the input voltage window. The voltage applied at the Data In connector needs to be within the voltage range defined by the common mode voltage and the voltage window.
This SCPI is supported by M8046A and all supported real-time oscilloscopes.

Example :INP:WIND 'M2.DataIn1', 0.4

:INPut[:VOLTage]:WINDow:ALLocate[?]
Syntax :INPut[:VOLTage]:WINDow:ALLocate 'Identifier', < NRf>
:INPut[:VOLTage]:WINDow:ALLocate? 'Identifier'
Input 'Identifier': 'INF*.DataIn1'
Parameters
Description This command defines how much of the input voltage window shall be used by the received signal.

This controls how the auto-set algorithm calculates the input voltage window size based on the measured signal amplitude. Normally the input voltage window is chosen larger than the signal amplitude to avoid nonlinear distortions at the peak voltage levels.

When working with small signal amplitudes, the BER results of the real-time oscilloscopes may be better if the received signal is using the input voltage range is used to a larger extent, than what the oscilloscope choses during the built-in vertical auto-scale operation.

Changing this parameter is not directly changing the input voltage range, but has effect on the next vertical auto-scale operation.

This SCPI is only supported by for real-time oscilloscopes.

Example :INP:WIND:ALL 'INF1.DataIn1', 90

\[
V_{\text{min}} = \text{Common Mode Voltage} - \frac{\text{Input Voltage Window}}{2}
\]

\[
V_{\text{max}} = \text{Common Mode Voltage} + \frac{\text{Input Voltage Window}}{2}
\]
:INPut[:VOLTage]:WINDow:DETect

Syntax :INPut[:VOLTage]:WINDow:DETect 'Identifier'

Input 'identifier': 'M*.DataIn'

Parameters

Description This command automatically adjust the input voltage window to match the received signal at the Data In connector.
- The execution progress can be queried using the command, see :INPut:ALIGnment:STATus[:VALue]? on page 277.
- The execution can be aborted using the command, see :INPut:ALIGnment:EYE:ABORt on page 273.

This SCPI is only applicable for M8046A.

Example :INP:VOLT:WIND:DET 'M2.DataIn'

:INPut[:VOLTage][:NRZ]:THReshold[?]?

Syntax :INPut[:VOLTage][:NRZ]:THReshold 'identifier', <NRf>

:INPut[:VOLTage][:NRZ]:THReshold? 'identifier'

Input <identifier>: Any valid data input identifier. M*.DataIn1 or M*.DataIn2

Parameters

Description The command specifies the threshold voltage of a RX input connector. This query returns the present setting.

Example :INP:THR 'M2.DataIn1', 0.2

:INPut[:VOLTage]:PAM4[:SYMBol]:THReshold(1|2|3)[?]?

Syntax :INPut[:VOLTage]:PAM4[:SYMBol]:THReshold(1|2|3) 'identifier', <NRf>

:INPut[:VOLTage]:PAM4[:SYMBol]:THReshold(1|2|3)? 'identifier'

Input <identifier>: Any valid data input identifier. M*.DataIn

Parameters

Description The command controls the decision threshold of the PAM4 decoder within the data input's input window. The setting of the PAM4 decision thresholds always refers to the actual input voltage range.

The 3 decision thresholds are applied as described in the following table:
This query returns the present state.

This SCPI is applicable for M8046A.

It is best practice to configure the 3 decision thresholds in a combined command transaction (see the example) to minimize hardware re-configuration duration.

Example:

```
:INP:PAM4:SYMB:THR1 'M2.DataIn',-100e-3;THR2 'M2.DataIn',-5e-3;THR3 'M2.DataIn',100e-3
```
:INPut:CDR Subnode

This subnode has the following SCPI structure:

```
:INPut
  CDR
   :LORDer[?]
   :CTRL[?]
   [:STATe][?]
   :AUTO[?]
   :RELOck
   :HIGHTD[?]
   :OPTimize
   :FIRSt
    :TDENsity[?]
    :LBANdwidth[?]
  :SECond
    :TDENsity[?]
    :LBANdwidth[?]
    :PEAKing[?]
    [:JTF]:SECond:LSELect:AUTO[?]
    [:JTF]:SECond:LSELect[:VALue][?]
  :OUTPut[?]
  :OJTF
   :STATe[?]
   :SECond[?]
    :LBANdwidth[?]
     :DIVisor[?]
     :DFACtor[?]
```
This subnode has the following commands:

Table 60

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:LORDer[?]</td>
<td>:INPut:CDR:LORDer[?] on page 263</td>
</tr>
<tr>
<td>:CTRL[?]</td>
<td>:INPut:CDR:CTRL[?] on page 263</td>
</tr>
<tr>
<td>[:STATe][?]</td>
<td>:INPut:CDR[:STATe][?] on page 264</td>
</tr>
<tr>
<td>:AUTO[?]</td>
<td>:INPut:CDR:AUTO[?] on page 264</td>
</tr>
<tr>
<td>:RELOck</td>
<td>:INPut:CDR:RELOck on page 265</td>
</tr>
<tr>
<td>:HIGHTD[?]</td>
<td>:INPut:CDR:HIGHTD[?] on page 265</td>
</tr>
<tr>
<td>OPTimize</td>
<td>:INPut:CDR:OPTimize on page 265</td>
</tr>
<tr>
<td>:FIRSt:TDENsity[?]</td>
<td>:INPut:CDR:FIRSt:TDENsity[?] on page 266</td>
</tr>
<tr>
<td>:FIRSt:LBANdwidth[?]</td>
<td>:INPut:CDR:FIRSt:LBANdwidth[?] on page 266</td>
</tr>
<tr>
<td>:INPut:CDR[:JTF][:STATe][?]</td>
<td>:INPut:CDR[:JTF][:STATe][?] on page 269</td>
</tr>
</tbody>
</table>

262 Keysight M8000 Series of BER Test Solutions Programming Guide
**:INPut:CDR:LORDer[?]**

**Syntax**

:INPut:CDR:LORDer 'identifier', <FIRSt|SECond>

:INPut:CDR:LORDer? 'identifier'

**Input**

'identifier': 'M*.DataIn1' or 'M*.DataIn2'

**Parameters**

<FIRSt|SECond>: Set the loop order of the CDR.

**Return Range**

FIR|SEC

**Description**

This command determines the loop order of the CDR. Depending on this setting, defining the behavior of the CDR is done by different parameters. For first order only, transition density and bandwidth fully define its characteristics while for the second order this definition needs at least one more parameter i.e. peaking.

This SCPI is applicable for M8041A and M8051A.

**Example**

:INP:CDR:LORD 'M1.DataIn1', FIR

**:INPut:CDR:CTRL[?]**

**Syntax**

:INPut:CDR:CTRL 'identifier', <MANual|SEQUence>

:INPut:CDR:CTRL? 'identifier'

**Input**

'identifier': 'M*.DataIn1' or 'M*.DataIn2'

**Parameters**

<MANual|SEQUence>: Set the CDR control method.

**Return Range**

MAN|SEQ

**Description**

This command selects whether the CDR is controlled manually or by the sequencer.

This SCPI is applicable for M8041A and M8051A.

**Example**

:INP:CDR:CTRL 'M1.DataIn1', SEQ
**:INPut:CDR[:STATe][?]**

**Syntax**  
:INPut:CDR[:STATe] 'identifier', <ON|OFF|1|0>  
:INPut:CDR[:STATe]? 'identifier'

**Input Parameters**  
- 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'
- <ON|OFF|1|0>: Enable/disable the CDR.

**Return Range**  
1|0

**Description**  
This command enables or disables CDR.

This SCPI is applicable for M8041A, M8051A and M8046A.

This command is only available as a query for M8046A.

**Example**  
:INP:CDR 'M1.DataIn1', ON


**:INPut:CDR:AUTO[?]**

**Syntax**  
:INPut:CDR:AUTO 'identifier', <ON|OFF|1|0>  
:INPut:CDR:AUTO? 'identifier'

**Input Parameters**  
- 'identifier': 'M*.DataIn'
- <ON|OFF|1|0>: Enable/disable the auto relock.

**Return Range**  
1|0

**Description**  
This command enables or disables the auto relock.

This SCPI is applicable for M8062A.

**Example**  
The following SCPI command sets the 'auto relock' to 1.
:INPut:CDR:AUTO 'M2.DataIn1', 1
The following SCPI command queries the 'auto relock' state.
:INPut:CDR:AUTO? 'M2.DataIn1'
1
:INPut:CDR:RELOck

Syntax   :INPut:CDR:RELOck 'identifier'
Input    'identifier': 'M*.DataIn'
Parameters
Description This command execute a full CDR relock.
This SCPI is applicable for M8062A.
Example   :INPut:CDR:RELOck 'M2.DataIn'

:INPut:CDR:HIGHTD[?]

Syntax   :INPut:CDR:HIGHTD 'identifier', <ON|OFF|1|0>
           :INPut:CDR:HIGHTD? 'identifier'
Input    'identifier': 'M*.DataIn'
Parameters <ON|OFF|1|0>: Set the High Transition Density state to ON or OFF.
Return Range 0|1
Description This command sets the High Transition Density state to ON or OFF.
This SCPI is applicable for M8062A.
Example   :INPut:CDR:HIGHTD 'M2.DataIn', 1

:INPut:CDR:OPTimize

Syntax   :INPut:CDR:OPTimize 'identifier'
Input    'identifier': 'M*.DataIn'
Parameters
Description This command invokes the M8062A CDR optimization at the current data rate and optionally applied jitter.
This SCPI is applicable for M8062A.
Example   :INPut:CDR:OPTimize 'M2.DataIn'
:INPut:CDR:FIRSt:TDENsity[?]

Syntax:
:INPut:CDR:FIRSt:TDENsity 'identifier', <NRf>
:INPut:CDR:FIRSt:TDENsity? 'identifier'

Input Parameters:

'identifier': 'M*.DataIn1' or 'M*.DataIn2'

<NRf>: Set first order transition density.

Return Range: 25% to 100%

Description: This command is used to set the first order transition density of the incoming signal from 25% to 100%. The default is 50%.

This SCPI is applicable for M8041A and M8051A.

Example:
:INP:CDR:FIRS:TDEN 'M1.DataIn1', 25

:INPut:CDR:FIRSt:LBANdwidth[?]

Syntax:
:INPut:CDR:FIRSt:LBANdwidth 'identifier', <NRf>
:INPut:CDR:FIRSt:LBANdwidth? 'identifier'

Input Parameters:

<NRf>: Set first order loop bandwidth of the CDR.

Return Range: 102 kHz to 20 MHz

Description: This command is used to set the first order loop bandwidth of the CDR. The default is 2 MHz. Acceptable units are Hz, kHz, MHz and exponents (for example, 4E6 is the same as 4 MHz).

This SCPI is applicable for M8041A and M8051A.

Example:
:INP:CDR:FIRS:LBAN 'M1.DataIn1', 4MHz
:INPut:CDR:SECond:TDENsity[?]  
Syntax  
:INPut:CDR:SECond:TDENsity 'identifier', <NRf>  
Input  
'identifier': 'M*.DataIn1' or 'M*.DataIn2'  
Parameters  
<NRf>: Set second order transition density.  
Return Range 25% to 100%  
Description This command is used to set the second order transition density of the incoming signal from 10% to 100%. The default is 50%. This SCPI is applicable for M8041A and M8051A.  
Example :INP:CDR:SEC:TDEN 'M1.DataIn1', 25

:INPut:CDR:SECond:LBANdwidth[?]  
Syntax  
:INPut:CDR:SECond:LBANdwidth 'identifier', <NRf>  
Input  
'identifier': 'M*.DataIn1' or 'M*.DataIn2'  
Parameters  
<NRf>: Set second order loop bandwidth of the CDR.  
Return Range 102 kHz to 20 MHz  
Description This command is used to set the second order loop bandwidth of the CDR. The default is 2 MHz. Acceptable units are Hz, kHz, MHz and exponents (for example, 4E6 is the same as 4 MHz). This SCPI is applicable for M8041A and M8051A.  
Example :INP:CDR:SEC:LBAN 'M1.DataIn1', 4MHz

:INPut:CDR:SECond:PEAKing[?]  
Syntax  
:INPut:CDR:SECond:PEAKing 'identifier', <NRf>  
Input  
'identifier': 'M*.DataIn1' or 'M*.DataIn2'  
Parameters  
<NRf>: Adjust second order peaking of the CDR.
Return Range 0 dB to 3 dB

Description This command adjusts the second order peaking of the CDR which is valid for the jitter transfer function. The default is 1 dB. This SCPI is applicable for M8041A and M8051A.

Example :INP:CDR:SEC:PEAK 'M1.DataIn1', 0.5

:INPut:CDR[:JTF]:SECond:LSELect[:VALue]

Syntax :INPut:CDR[:JTF]:SECond:LSELect[:VALue] 'identifier', <LOOP1 | LOOP2 | LOOP3 | LOOP4>

:INPut:CDR[:JTF]:SECond:LSELect[:VALue]? 'identifier'

Identifier 'M*.DataIn'

Description This command specifies 2 loop transition frequency and thereby the JTF peaking.

This query returns the present setting.

This SCPI is applicable for M8146A.

Example :INP:CDR:SEC:LSEL 'M2.DataIn', LOOP2

:INPut:CDR:OUTPut[?]

Syntax :INPut:CDR:OUTPut 'identifier', <ON|OFF|1|0>

:INPut:CDR:OUTPut? 'identifier'

Identifier 'M*.DataIn'

Description This command enables or disables the recovered clock signals at the Recovered Clock Output.

This query returns the present setting.

This SCPI is applicable for M8146A.

Example :INP:CDR:OUTP 'M2.DataIn', ON
The :INPut:CDR:OJTF subnode allows controlling the clock recovery on an error detector channel that is based on a real-time oscilloscope.

**:INPut:CDR:OJTF[:STATe]

Syntax
:INPut:CDR:OJTF[:STATe] <identifier>,{OFF | ON | 0 | 1}
:INPut:CDR:OJTF[:STATe]? <identifier>

Identifier 'INF*.DataIn1'

Description
This command enables or disables the use of the clock recovery. For the current versions of M8070B, the clock recovery will always be enabled and cannot be disabled.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example
:INP:CDR:OJTF:STAT 'INF1.DataIn1',ON

**:INPut:CDR:OJTF:SECond:LBANdwidth

Syntax

Identifier 'INF*.DataIn1'

Description
This command configures the loop bandwidth of the clock recovery. For more details, refer the corresponding command in the Keysight Infiniium Oscilloscopes Programmer’s Guide.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example


Identifier 'INF*.DataIn1'

Description This command configures the loop bandwidth of the CDR by a symbol rate divisor. The resulting loop bandwidth will be the channel’s symbol rate divided by the loop bandwidth divisor.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.


Identifier 'INF*.DataIn1'

Description This command configures the damping factor of the clock recovery. For more details, refer the corresponding command in the Keysight Infiniium Oscilloscopes Programmer’s Guide.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example :INP:CDR:OJTF:SEC:DFAC 'INF1.DataIn1',0.707
:INPut:ALIGNment Subnode

This subnode has the following SCPI structure:

```
:INPut
  :ALIGNment
    :EYE
      [:AUTO]
      :ACENter
      :TCENter
      :ABORT
      :THReshold[?]
    :RESult
      [:NRZ][:HEIGth]? 
      :WIDTH?
      [:NRZ]:THReshold?
      :PAM4:HEIGht(1|2|3)?
      :PAM4:THReshold(1|2|3)?
      :DELay?
      :POLarity?
    :STATus[:VALue]?
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:EYE:RESult[:NRZ][HEIGht]?</td>
<td>:INPut:ALIGNment:EYE:RESult[:NRZ][HEIGht]? on page 274</td>
</tr>
</tbody>
</table>
### :INPut:ALIgnment:EYE[:AUTO]

**Syntax**

`:INPut:ALIgnment:EYE[:AUTO] 'identifier'`

**Input Parameters**

'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

**Description**

This command starts an auto alignment.

This SCPI is applicable for M8041A, M8051A, M8061A, M8046A and real-time oscilloscope based error detector channels.

**Example**

`:INP:ALIG:EYE 'M1.DataIn1'`

### :INPut:ALIgnment:EYE:ACEN

**Syntax**

`:INPut:ALIgnment:EYE:ACEN 'identifier'`

**Input Parameters**

'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

**Description**

This command initiates a search for the 0/1 threshold voltage midway between the two 0/1 threshold voltages with a measured BER just in excess of the BER configured by the :EYE:THReshold command. If successful, the command leaves the 0/1 threshold at this value.

This SCPI is applicable for M8041A, M8051A, M8061A, M8046A and real-time oscilloscope based error detector channels.

**Example**

`:INP:ALIG:EYE:ACEN 'M1.DataIn1'`

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
</table>
:INPut:ALIGnment:EYE:TCEN

Syntax  :INPut:ALIGnment:EYE:TCEN 'identifier'

Input   'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

Parameters

Description  This command initiates a search for the value of data/clock delay that puts
the active clock edge in the center of the data eye, midway between the
two relative delay points with a measured BER just in excess of the BER
configured by the :EYE:THReshold command. If successful, the command
leaves the data/clock delay at this value.

This SCPI is applicable for M8041A, M8051A, M8061A, M8046A and
real-time oscilloscope based error detector channels.

Example  :INP:ALIG:EYE:TCEN 'M1.DataIn1'

:INPut:ALIGnment:EYE:ABORt

Syntax  :INPut:ALIGnment:EYE:ABORt 'identifier'

Input   'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

Parameters

Description  This command interrupts and aborts an alignment.

This SCPI is applicable for M8041A, M8051A, M8061A, M8046A and
real-time oscilloscope based error detector channels.

Example  :INP:ALIG:EYE:ABOR 'M1.DataIn1'
**:INPut:ALIgnment:EYE:THReshold[?]**

**Syntax**

:INPut:ALIgnment:EYE:THReshold 'identifier', <NRf>

:INPut:ALIgnment:EYE:THReshold? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'
- <NRf>: Set BER threshold.

**Return Range**

1E-9 to 1E-1

**Description**

The command sets the BER threshold to be used in the determination of the edges of the eye. The query returns the current BER threshold value.

This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

**Example**

:INP:ALIG:EYE:THR 'M1.DataIn1', 1E-3

**:INPut:ALIgnment:EYE:RESult[:NRZ][:HEIGht]??**

**Syntax**

:INPut:ALIgnment:EYE:RESult[:NRZ][:HEIGht]? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

**Description**

This is a query command that searches for the value of data amplitude that puts the 0/1 threshold level midway between the upper and lower bounds at which the error ratio exceeds the threshold value set by the :INPut:ALIgnment:EYE:THReshold command.

This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

**Example**

:INP:ALIG:EYE:RES? 'M1.DataIn1'

**:INPut:ALIgnment:EYE:RESult:WIDTh??**

**Syntax**

:INPut:ALIgnment:EYE:RESult:WIDTh? 'identifier'

**Input Parameters**

- 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

**Description**

This is a query command that interrogates the eye width found by the most recent search for the value of data/clock delay that put the active edge in the center of the data eye.
This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.


:INP:ALigNment:EYE:RESult[:NRZ]:THReshold?
Syntax: :INP:ALigNment:EYE:RESult[:NRZ]:THReshold? 'identifier'
Input: 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'
Parameters: Returns the threshold detected at the last alignment run.
Description: This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

:INP:ALigNment:EYE:RESult:PAM4:HEIGht(1|2|3)?
Syntax: :INP:ALigNment:EYE:RESult:PAM4:HEIGht(1|2|3)? 'identifier'
Input: 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'
Parameters: This query returns the value of data amplitude that puts the 0/1 threshold level midway between the upper and lower bounds at which the error ratio exceeds the threshold value set by the :INP:ALigNment:EYE:THReshold command.
The suffix addresses the height of the lower eye (1), middle eye (2) or upper eye (3). If the suffix is omitted, then the returned value is for the lower eye of the PAM4 signal.
This SCPI is applicable for M8046A.

:INP:ALigNment:EYE:RESult:PAM4:THReshold(1|2|3)?
Syntax: :INP:ALigNment:EYE:RESult:PAM4:THReshold(1|2|3)? 'identifier'
Input: 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'
Parameters: This query returns the threshold detected at the last alignment run. The suffix addresses the threshold of the lower eye (1), middle eye (2) or upper eye (3). If the suffix is omitted, then the returned value is for the lower eye
of the PAM4 signal.
This SCPI is applicable for M8046A.


:INPut:ALIGNment:EYE:RESult:DElay?


Input 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

Input Parameters

Description Returns the delay detected at the last alignment run.
This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

Example :INP:ALIG:EYE:RES:DEL? 'M1.DataIn1'

:INPut:ALIGNment:EYE:RESult:POLarity?


Input 'identifier': 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

Return Range NOR|INV

Description Returns the polarity of the signal detected at the last alignment run.
This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

Example :INP:ALIG:EYE:RES:POL? 'M1.DataIn1'
:INPut:ALIGNment:STATus Subnode

This subnode reports alignment specific status information and offers an alternative way of retrieving the same status information that is available via the :STATus:INSTrument:ALIGNment Subnode on page 130.

:INPut:ALIGNment:STATus[:VALue]? Syntax

:INPut:ALIGNment:STATus[:VALue]? 'Identifier'

Input

'identifier': ‘Any error detector location and group name. E.g. 'M1.DataIn1', 'M2.DataIn', 'INF1.DataIn1'

Parameters

Description

This command retrieves alignment status information for an error detector or error detector group.

This command can be used to retrieve information about the alignment procedure that is working on the addressed error detector channel, or a group of error detectors. This command is reporting the status independently of the executing alignment procedure. There is no differentiation in respect to different types of alignments.

This command returns the following values:

- **INPROGRESS** - At least one addressed error detector is executing an alignment procedure.
- **ABORTED** - The alignment procedure has been aborted by the user before all aligning error detector channels finished the procedure.
- **FAILED** - All addressed error detector channels have finished the alignment procedure, but at least one was unable to finish successfully.
- **SUCCESS** - The addressed error detector channels have successfully finished the alignment procedure. Error detector channels that are addressed by the command but return UNKNOWN state, do not contribute to the result.
- **UNKNOWN** - None of the addressed error detectors has executed an alignment procedure yet.
This command cannot be used in the same compound SCPI statement that is starting the alignment procedure. If this command is sent to the instrument right after the command that is initiating the alignment procedure, then the returned value is undefined and can reflect the state that was valid before the alignment. This can be overcome by reading any value from the instrument (e.g. *ESE?), except for *OPC? as this will block until the alignment procedure is finished.

Example :INP:ALIGN:STAT? 'M1.DataIn1'
This subnode controls the equalizer settings on a data input. Depending on the product number different methods of equalizations are available.

**:INPut:EQUalization**

This equalizer can operate in the following three modes:

- **Equalizer Presets with cable compensation**
  
  The equalizer gain is controlled as Equalizer Level, and the losses of the standard cabling is automatically included in the equalizer setting.

- **Manual coefficient entry with cable compensation**
  
  The coefficients of the Feed Forward Equalizer can be entered manually. Additionally, the losses of the standard cabling is automatically included in the equalizer setting.

- **Manual coefficient entry**
  
  The coefficients of the Feed Forward Equalizer can be entered manually. No cable losses are automatically compensated. Use this mode when using non-standard cabling.

Additional to the manual control of the equalizer mode and coefficient settings, the coefficients can be automatically determined from the received signal.
This subnode has the following SCPI structure:
This subnode has the following commands:

### Table 62

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:INPut:EQUalization:CURSor:MAGNitude[0</td>
<td>1</td>
</tr>
<tr>
<td>:INPut:EQUalization[:PREset][?]</td>
<td>:INPut:EQUalization[:PREset][?] on page 284</td>
</tr>
</tbody>
</table>
:INPut:EQUalization:CURSor:MODE[?]  

**Syntax:**  
```
:INPut:EQUalization:CURSor:MODE 'Identifier', {PRESet | MCABle | MANual}
```

```
:INPut:EQUalization:CURSor:MODE? 'Identifier'
```

**Input Parameters:**
- `'identifier': 'M*.DataIn'
- PRESet | MCABle | MANual

**Description:** This command the equalization control and behavior. The following three equalization modes are available for operation:

- **PRESet**
  - Select the EQ gain level via presets with automatic cable compensation.

- **MCABle**

- **MANual**
  - Manual entry of FFE coefficients without cable compensation.

This SCPI is only applicable for M8046A.

**Example:**
```
:INP:EQU:CURS:MODE 'M2.DataIn', MAN
```
:INPut:EQUalization:CURSor:MAGNitude[0|1|2|3|...|15][?]

Syntax: :INPut:EQUalization:CURSor:MAGNitude[0|1|2|3|...|15] 'Identifier', <NRf>
:INPut:EQUalization:CURSor:MAGNitude[0|1|2|3|...|15]? 'Identifier'

Input Parameters:
- 'identifier': 'M*.DataIn'

Description:
This command allows to set the equalizer coefficients manually. There are 16 filter coefficients, numbered from 0 to 15. Coefficient 2 is the main-cursor and cannot be changed.

The available value range depends on the coefficient and is defined as follows:
- Coefficient 0: -0.25 .. + 0.25
- Coefficient 1: -0.5 .. + 0.5
- Coefficient 2: 1.0
- Coefficient 3: -0.5 .. + 0.5
- Coefficient 4: -0.25 .. + 0.25
- Coefficient 5: -0.125 .. +0.125
- Coefficient 6 to 15: -0.0625 .. +0.0625

Additionally, the sum of all 16 coefficients may not be 0.

This SCPI is only applicable for M8046A.

Example:
:INP:EQU:CURS:MAGN3 'M2.DataIn', -0.15
:INP:EQU:CURS:MAGN3? 'M2.DataIn' -> -0.15
:INPut:EQUalization:CURYsor:DETECT
Syntax :INPut:EQUalization:CURYsor:DETECT 'Identifier'
Input 'identifier': 'M*.DataIn'
Parameters
Description This command automatically determines the correct FFE coefficients for
the current input signal.
- The execution progress can be queried using the command, see
- The execution can be aborted using the command, see
This SCPI is only applicable for M8046A.
Example :INP:EQU:CURS:DET 'M2.DataIn'

:INPut:EQUalization[:PREset][:?]  
Syntax :INPut:EQUalization[:PREset] 'Identifier', <OFF | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10>
:INPut:EQUalization[:PREset]? 'Identifier'
Input 'identifier': 'M*.DataIn1' or 'M*.DataIn2'
Parameters <OFF | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10>
Return Range OFF | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10
Description This command is used to select from the following equalization presets:
For M8041A and M8051A:
- OFF - Disables equalization
- P1 - PCIe 8Gb/s -6 dB
- P2 - PCIe 8Gb/s -9 dB
- P3 - PCIe 8Gb/s -12 dB
- P4 - USB 5Gb/s -3.5 dB
- P5 - PCIe 16Gb/s -6 dB
- P6 - PCIe 16Gb/s -9 dB
- P7 - PCIe 16Gb/s -12 dB
- P8 - USB 10Gb/s 0 dB
- P9 - USB 10Gb/s -3 dB
- P10 - USB 10Gb/s -6 dB
For M8061A:
- OFF - off
- P1 - PCIe 8Gb/s -6 dB
- P2 - PCIe 8Gb/s -9 dB
- P3 - PCIe 8Gb/s -12 dB
- P4 - USB 5Gb/s -3.5 dB

For M8062A:
- P1 - Low
- P2 - Medium
- P3 - High

The availability of individual presets depends on the type of module being controlled and the calibration status of the respective module.

This SCPI is applicable for M8041A, M8051A, M8061A and M8062A.

For M8062A, the equalizer is available only for 20 Gb/s and higher. For other values, if a command is sent, it will return OFF and will notify that the value is read-only.

Example: :INP:EQU 'M1.DataIn1',P1

:INPut:EQUalization[:PRESet]:LEVel[?] Syntax :INPut:EQUalization[:PRESet]:LEVel 'Identifier', <NRf>
:INPut:EQUalization[:PRESet]:LEVel ? 'Identifier'

Input 'Identifier': 'M*.DataIn'
Parameters The specified number has a range from 0 – 120.
Return Range 0 – 120
Description This command controls the integrated and adjustable equalization of M8046A, to compensate for loss characteristic of the back channel. This feature is available for NRZ and PAM4 signals.

Equalization can be adjusted from no equalization (0) to maximum equalization (120) in equidistant steps. The available range is restricted to 0 – 55 for PAM4 signals.

This SCPI is only applicable for M8046A.
Example: `:INP:EQU:LEV 'M1.DataIn', 12`
`:INP:EQU:LEV? 'M1.DataIn'`

`:INPut:EQUalization:FFEQualizer Subnode

This subnode configures a Feed-Forward Equalizer on a data input. This subnode is available for real-time oscilloscope based error detector channels. It requires the existence for the Equalization license (option DEQ) on the real-time oscilloscope. If the license is not available, then equalization will not be offered in M8070B for the data input.

`:INPut:EQUalization:FFEQualizer:STATe[?]

Syntax: `:INPut:EQUalization:FFEQualizer:STATe <identifier>,{OFF|ON|0|1}`
`:INPut:EQUalization:FFEQualizer:STATe? <identifier>`

Identifier: `'INF*.DataIn1'`

Description: This command enables or disables the use of the FFE algorithm on the given data input.
This query returns the present setting.
This SCPI is applicable for real-time oscilloscope based error detector channels.

Example: `:INP:EQU:FFEQ:STAT 'INF1.DataIn1', ON`

`:INPut:EQUalization:FFEQualizer:NTAPs[?]

Syntax: `:INPut:EQUalization:FFEQualizer:NTAPs <identifier>,<number>`
`:INPut:EQUalization:FFEQualizer:NTAPs? <identifier>`

Identifier: `'INF*.DataIn1'`

Description: This command sets the number of taps to be used in the FFE algorithm.
This query returns the present setting.
This SCPI is applicable for real-time oscilloscope based error detector channels.

Example: `:INP:EQU:FFEQ: NTAPs 'INF1.DataIn1',6`
**:INPut:EQUalization:FFEQualizer:NPRecursor[?]**

**Syntax**: 
:INPut:EQUalization:FFEQualizer:NPRecursor <identifier>,<number>

**Identifier**: 'INF*.DataIn1'

**Description**: This command sets the number of precursor taps to be used in the FFE algorithm.

- This query returns the present setting.
- This SCPI is applicable for real-time oscilloscope based error detector channels.

**Example**: 
:INP:EQU:FFEQ:NPR 'INF1.DataIn1',2

**:INPut:EQUalization:FFEQualizer:TAP:AUTO[?]**

**Syntax**: 
:INPut:EQUalization:FFEQualizer:TAP:AUTO <identifier>

**Identifier**: 'INF*.DataIn1'

**Description**: This command starts the FFE tap optimization. Be sure to first specify the number of taps.

- This query returns the present setting.
- This SCPI is applicable for real-time oscilloscope based error detector channels.

**Example**: 
:INP:EQU:FFEQ:TAP:AUTO 'INF1.DataIn1'

**:INPut:EQUalization:FFEQualizer:TAP:AUTO:INCLude[?]**

**Syntax**: 
:INPut:EQUalization:FFEQualizer:TAP:AUTO:INCLude <identifier>, <0|1|OFF|ON>

**Identifier**: 'INF*.DataIn1'
Description

This command controls if the FFE tap optimization is done during a full sample point alignment. If this is set to OFF, then the tap coefficients must be set manually, or by executing the :INPut:EQUalization:FFEQualizer:TAP:AUTO command.

FFE tap optimization is part of the full sample point optimization by default. Use this command to disable tap optimization in scenarios where the input signal is too distorted to produce reliable tap optimization results.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example

:INP:EQU:FFEQ:TAP:AUTO:INCL 'INF1.DataIn1', ON
ON

:INPut:EQUalization:FFEQualizer:TAP:WIDTh[?]

Syntax

:INPut:EQUalization:FFEQualizer:TAP:WIDTh <identifier>,<width>

Identifier

'INF*.DataIn1'

Description

The tap width sets the Eye Width field for the FFE tap optimization. Setting the width to 0.0 means the optimization is only performed at the location of the clock. Setting the width to 1.0 means the entire acquisition is used in the optimization. The default value for FFE is 0.33. For more information on this parameter, refer to the N5461A Infiniium Serial Data Equalization User’s Guide

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example

:INP:EQU:FFEQ:TAP:WIDT 'INF1.DataIn1',0.1
:INP:EQU:FFEQ:TAP:WIDT? 'INF1.DataIn1' -> 0.1
The CTLequalizer subsystem is configuring the use of the Constant Time Linear Equalizer (CTLE) of a real-time oscilloscope that is used as error detector in M8070B.

**:INPut:EQUalization:CTLequalizer:ACGain[?]**

**Syntax**

```
:INPut:EQUalization:CTLequalizer:ACGain <identifier>,<NRf>
```

**Identifier**

'INF*.DataIn1'

**Input Parameters**

```
<identifier>
<NRf> - AC Gain
```

**Description**

This command sets the AC Gain parameter for the Continuous Time Linear Equalization when '2-poles with AC gain' is selected for the "# of Poles" parameter.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

**Example**

```
:INP:EQU:CTL:ACG 'INF1.DataIn1',1
```

**:INPut:EQUalization:CTLequalizer:DCGain[?]**

**Syntax**

```
:INPut:EQUalization:CTLequalizer:DCGain <identifier>,<NRf>
 :INPut:EQUalization:CTLequalizer:DCGain? <identifier>
```

**Identifier**

'INF*.DataIn1'

**Input Parameters**

```
<identifier>
<NRf> - DC Gain
```

**Description**

This command sets the DC Gain parameter for the Continuous Time Linear Equalization.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

**Example**

```
:INP:EQU:CTL:DCG 'INF1.DataIn1',1
```
:INPut:EQUalization:CTLequalizer:STATe[?]

Syntax :INPut:EQUalization:CTLequalizer:STATe <identifier>, {OFF | ON | 0 | 1}


Identifier 'INF*.DataIn1'

Input <identifier>

Parameters OFF | ON | 0 | 1

Description This command enables or disables the use of the Continuous Time Linear Equalization.

The CTLE cannot be used at the same time as the Feed Forward Equalization (FFE).

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example :INP:EQU:CTL:STAT 'INF1.DataIn1',ON

:INPut:EQUalization:CTLequalizer:NUMPoles[?]

Syntax :INPut:EQUalization:CTLequalizer:NUMPoles <identifier>, { P2Z1 | P3Z1 | P3Z2 | P2ACG }


Identifier 'INF*.DataIn1'

Input <identifier>

Parameters P2Z1 | P3Z1 | P3Z2 | P2ACG

Description This command selects either a 2 Pole 1 Zero, 3 Pole 1 Zero, 3 Pole 2 Zeros, or 2 Poles and AC gain (for USB 3.1) Continuous Time Linear Equalizer (CTLE).

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example :INP:EQU:CTL:NUMP 'INF1.DataIn1',P3Z1
:INPut:EQUalization:CTLequalizer:P1[?]

Syntax :INPut:EQUalization:CTLequalizer:P1 <identifier>, <NRf>
         :INPut:EQUalization:CTLequalizer:P1? <identifier>

Identifier 'INF*.DataIn1'

Input <identifier>

Parameters <NRf> - Pole 1 frequency

Description This command sets the Pole 1 frequency for the Continuous Time Linear Equalization.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example :INP:EQU:CTL:P1 'INF1.DataIn1',1e9

:INPut:EQUalization:CTLequalizer:P2[?]

Syntax :INPut:EQUalization:CTLequalizer:P2 <identifier>, <NRf>

Identifier 'INF*.DataIn1'

Input <identifier>

Parameters <NRf> - Pole 2 frequency

Description This command sets the Pole 2 frequency for the Continuous Time Linear Equalization.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example :INP:EQU:CTL:P2 'INF1.DataIn1',4e9

:INPut:EQUalization:CTLequalizer:P3[?]

Syntax :INPut:EQUalization:CTLequalizer:P3 <identifier>, <NRf>

Identifier 'INF*.DataIn1'
Input Parameters

Input:INPut:EQUalization:CTLequalizer:Z1 <identifier>, <NRf>

Description

This command sets first zero frequency for the 2-pole and 3-pole Continuous Time Linear Equalization.

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example

:INP:EQU:CTL:Z1 'INF1.DataIn1',650e6

:INPut:EQUalization:CTLequalizer:Z2 <identifier>, <NRf>

Description

This command sets second zero frequency for the 2-pole and 3-pole Continuous Time Linear Equalization.
This query returns the present setting.
This SCPI is applicable for real-time oscilloscope based error detector channels.

Example: :INP:EQU:CTL:Z2 INF1.DataIn1",4e9

:INPut:EQUalization:CTLequalizer:PRESet[?]


Identifier: 'INF*.DataIn1'

Input Parameters:

- "preset-name" - preset name, see the supporting preset names.

Supported preset names are:
- "CEI-56G-VSR-NRZ (1 dB)"
- "CEI-56G-VSR-NRZ (2 dB)"
- "CEI-56G-VSR-NRZ (3 dB)"
- "CEI-56G-VSR-NRZ (4 dB)"
- "CEI-56G-VSR-NRZ (5 dB)"
- "CEI-56G-VSR-NRZ (6 dB)"
- "CEI-56G-VSR-NRZ (7 dB)"
- "CEI-56G-VSR-NRZ (8 dB)"
- "CEI-56G-VSR-NRZ (9 dB)"
- "CEI-56G-VSR-NRZ (10 dB)"
- "CEI-56G-VSR-NRZ (11 dB)"
- "CEI-56G-VSR-NRZ (12 dB)"
- "IEEE 802.3bs CDAUI-8 (1 dB)"
- "IEEE 802.3bs CDAUI-8 (1.5 dB)"
- "IEEE 802.3bs CDAUI-8 (2 dB)"
- "IEEE 802.3bs CDAUI-8 (2.5 dB)"
- "IEEE 802.3bs CDAUI-8 (3 dB)"
- "IEEE 802.3bs CDAUI-8 (3.5 dB)"
- "IEEE 802.3bs CDAUI-8 (4 dB)"
- "IEEE 802.3bs CDAUI-8 (4.5 dB)"
- "IEEE 802.3bs CDAUI-8 (5 dB)"
- "IEEE 802.3bs CDAUI-8 (5.5 dB)"
- "IEEE 802.3bs CDAUI-8 (6 dB)"
- "IEEE 802.3bs CDAUI-8 (6.5 dB)"
- "IEEE 802.3bs CDAUI-8 (7 dB)"
- "IEEE 802.3bs CDAUI-8 (7.5 dB)"
- "IEEE 802.3bs CDAUI-8 (8 dB)"
- "IEEE 802.3bs CDAUI-8 (8.5 dB)"
- "IEEE 802.3bs CDAUI-8 (9 dB)"
- "OIF CEI-28G-VSR (0 dB)"
- "OIF CEI-28G-VSR (1 dB)"
- "OIF CEI-28G-VSR (2 dB)"
- "OIF CEI-28G-VSR (3 dB)"
- "OIF CEI-28G-VSR (4 dB)"
- "OIF CEI-28G-VSR (5 dB)"
- "OIF CEI-28G-VSR (6 dB)"
- "OIF CEI-28G-VSR (7 dB)"
- "OIF CEI-28G-VSR (8 dB)"
- "OIF CEI-28G-VSR (9 dB)"
- "PCIe Gen3 (6 dB)"
- "PCIe Gen3 (7 dB)"
- "PCIe Gen3 (8 dB)"
- "PCIe Gen3 (9 dB)"
- "PCIe Gen3 (10 dB)"
- "PCIe Gen3 (11 dB)"
- "PCIe Gen3 (12 dB)"
- "PCIe Gen4 (6 dB)"
- "PCIe Gen4 (7 dB)"
- "PCIe Gen4 (8 dB)"
- "PCIe Gen4 (9 dB)"
- "PCIe Gen4 (10 dB)"
- "PCIe Gen4 (11 dB)"
- "PCIe Gen4 (12 dB)"
- "PCIe Gen5 (6 dB)"
- "PCIe Gen5 (7 dB)"
- "PCIe Gen5 (8 dB)"
Description
This command selects a predefined setting of the Continuous Time Linear Equalizer (CTLE).

This query returns the present setting.

This SCPI is applicable for real-time oscilloscope based error detector channels.

Example
:INP:EQU:CTL:PRES 'INF1.DataIn1', "PCIe Gen3 (6dB)"
The Squelch Threshold feature allows you to adjust the squelch threshold values in link training applications (e.g. USB link training).

This subnode has the following SCPI structure:

```
:INPut:SQUElch
   :MODe[?]
   :VALue[?]
   :PREset
```

This subnode has the following commands:

Table 63

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
</table>

**:INPut:SQUElch:MODe[?]**

Syntax: :INPut:SQUElch:MODe <identifier>, {MANual | AUTo}

:INPut:SQUElch:MODe <identifier>

Identifier: 'M*.DataIn1' or 'M*.DataIn2'

Description: This command selects the squelch mode to either manual or auto on a data input.

This query returns the present setting.

This SCPI is applicable for M8041A and M8051A.

Example:

```
:INP:SQUE:MOD 'M2.DataIn2',AUT
:INP:SQUE:MOD? 'M2.DataIn2' -> AUT
```
:INPut:SQUElich:VALue[?]  
**Syntax**: :INPut:SQUElich:VALue <identifier>,<number>  
:INPut:SQUElich:VALue? <identifier>  
**Identifier**: 'M*.DataIn1' or 'M*.DataIn2'  
**Description**: This command sets the squelch threshold value in range from 0 ... 255.  
This query returns the present setting.  
This SCPI is applicable for M8041A and M8051A.  
**Example**: :INP:SQUE:VAL 'M2.DataIn2',40  
:INP:SQUE:VAL? 'M2.DataIn2' -> 40

:INPut:SQUElich:VALue:PREset  
**Syntax**: :INPut:SQUElich:VALue:PREset <identifier>  
**Identifier**: 'M*.DataIn1' or 'M*.DataIn2'  
**Description**: This command sets the default squelch threshold value.  
This SCPI is applicable for M8041A and M8051A.  
**Example**: :INP:SQUE:VAL:PRE 'M2.DataIn2'
SENSe Subsystem

The SENSe subsystem represents the analyzer's Data In port. This subsystem has the following commands:

Table 64

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SENSe:SRATe[?]?</td>
<td>:SENSe:SRATe[?]? on page 298</td>
</tr>
</tbody>
</table>

:SENSe:SRATe[?]

Syntax :SENSe:SRATe <identifier>, <NRF>
:SENSe:SRATe? <identifier>

Input Parameters

- 'identifier': 'M*.DataIn', 'INF*.DataIn1'

Description

This command controls the symbol rate for the data input. The symbol rate can only be controlled directly when the clock source is set to CDR. When Clk In or Sys Clk is selected as the clock source, then the symbol rate can only be read.

This query returns the present setting.

This SCPI is applicable for M8046A and real-time oscilloscope based error detector channels.

Example :SENSe:SRAT  'M2.DataIn1', 10e9
The CLOCk subsystem is used to detect clock source or to set the clock frequency applied to input or output channel.

**NOTE**
M8061A has only DataOut with Clock functional block, so the commands are only supported in the DataOut location and not in the DataIn location.

This subsystem has the following SCPI structure:

```
:CLOCk
 [:SOURce][?]
[:CRECovery:SELect][?]
:FREQuency[?]
[:MULTiplier][?]
:TRACk
 [:STATe][?]
```

This subsystem has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:SOURce][?]</td>
<td>:CLOCk [:SOURce][?] on page 300</td>
</tr>
<tr>
<td>:CLOCk[:SOURce]:CRECovery:SELect[?]</td>
<td>:CLOCk[:SOURce]:CRECovery:SELect[?] on page 300</td>
</tr>
<tr>
<td>:FREQuency[?]</td>
<td>:CLOCk[:SOURce]:CRECovery:SELect[?] on page 300</td>
</tr>
<tr>
<td>:FREQuency:MULTiplier[?]</td>
<td>:CLOCk:FREQuency:MULTiplier[?] on page 301</td>
</tr>
<tr>
<td>:TRACk[:STATe][?]</td>
<td>:CLOCk:TRACk [:STATe][?] on page 302</td>
</tr>
</tbody>
</table>
:CLOCKSOURce]? 
Syntax :CLOCk[SOURce] 'identifier', <NRf> 
 :CLOCk[SOURce]? 'identifier' 
Input 'identifier': 'M*.DataOut' and 'M*.DataIn' 
Parameters <NRf>: CDR | CLK | SYS | AUXCLK | ECR
Return Range CDR | CLK | SYS | AUXCLK | ECR
Description This command sets the source of the clock signal. It is supported by: 
• M8061A 
• M8062A 
• M8046A 
The availability of the different clock sources depends on the product being controlled and possible even on installed product options (e.g. CDR). The following are the different clock sources that can be selected: 

CDR - Clock Recovery 
CLK - Front panel 'CLK IN' connector 
SYS - Internal clock generation 
AUXCLK - Front panel 'AUX CLK IN' connector 
ECCrecovery - External clock recovery. Selection is done with the command :CLOCk[SOURce]:CRECovery:SELect. For details, see :CLOCk[SOURce]:CRECovery:SELect[?] on page 300. 
This query returns the present state. 
This SCPI is applicable for M8061A, M8062A and M8046A. 
Example :CLOCk:SOURce 'M3.DataIn', CLK 

:CLOCk[SOURce]:CRECovery:SELect[?] 
Syntax :CLOCk:CRECovery 'Identifier', 'Clock Recovery Identifier' 
:CLOCk:CRECovery? 'Identifier' 
Input 'identifier': 'M*.DataIn' 
Parameters 'Clock Recovery Identifier' - ECR 
Return Range ECR
Description
This command select the external clock recovery to be used when using the clock source 'ECRecovery'.
This query returns the present state.
This SCPI is applicable for M8046A.
Example
:CLOCK:CREC:SEL 'M2.DataIn', 'DCA1.Slot1'

:CLOCK:FREQuency [?]
Syntax
CLOCK:FREQuency 'Identifier', <NRf>
CLOCK:FREQuency? 'Identifier'
Input Parameters
'identifier': 'M*.DataOut' or 'M*.DataIn'
Return Range NRf
Description
This command sets the clock frequency applied to the output channel.
This query returns the present state.
This SCPI is applicable for M8061A, M8062A and M8046A.
Example
:CLOCK:FREQ 'M3.DataOut', 8.0e9

:CLOCK:FREQuency:MULTiplier[?]
Syntax
:CLOCK:FREQuency:MULTiplier 'Identifier', <NRf>
:CLOCK:FREQuency:MULTiplier? 'Identifier'
Input Parameters
'identifier': 'M*.DataIn'
Return Range NRf
Description
This command sets the frequency multiplier that is applied to the externally provided clock in order to achieve the desired symbol rate.
Symbol Rate = Multiplier x Clock Frequency
This query returns the present setting.
This SCPI is applicable for M8046A.
Example
:CLOCK:FREQ:MULT 'M2.DataIn',2
**:CLOCk:TRACk[:STATe][?]**

**Syntax**

`:CLOCk:TRACk[:STATe] <identifier>, <ON|OFF|1|0>

`:CLOCk:TRACk[:STATe]? 'Identifier'

**Input**

'identifier': 'M*.DataIn', 'INF*.DataIn1'

**Parameters**

**Description**

This command enables or disables the automatic system symbol rate tracking. When this is enabled, then the system symbol rate is used as the symbol rate of the addressed functional unit (e.g. DataIn). In this case, the required clock frequency will be determined automatically based on the selected frequency multiplier.

See:
- `:CLOCk:FREQuency:MULTiplier[?]`
- `:[SENSe]:SRATe[?]`

This query returns the present state.

This SCPI is applicable for M8046A and real-time oscilloscope based error detector channels.

**Example**

`:CLOC:TRAC 'M2.DataIn', ON`
N1000 Subsystem

The N1000 subsystem is used to control external clock recovery instruments that are themselves controlled from within the FlexDCA N1000-Series System Software.

The minimum required version of the FlexDCA N1000-Series System Software is A.05.61.23.

It is possible to share one FlexDCA instance among multiple M8070B instances. However, to ensure correct access control, all M8070B instances have to be executed by the same PC.

All the commands used in this subsystem are basically identical with the SCPI commands of FlexDCA N1000-Series System Software. The only difference is that the M8070B does not use a suffix to address specific parts of the instrument, but instead always has the identifier parameter as first argument of the SCPI commands.

Detailed information about the features and SCPI commands of the supported clock recoveries can be found in the corresponding documentation:

**N1076A User Guide**

**N1010A FlexDCA Online Help**

**N1076B Documentation**

**N1078A Documentation**

**CDR Data Sheet**
This subsystem has the following SCPI structure:

```
:N1000
  :EMODules
    :CONNect[?]   :N1000:EMODules:SLOT:CONNect[?] on page 305
    :SLOT:MODEl?
    :CRECovery
      :CLBandwidth[?]
      :CRATe [?]
      :ELEVel[?]
      :LBWMode[?]
      :LOCKed?
      :LSELect [?]
      :AUTomatic[?]
      :ODRatio[?]
      :AUTO[?]
      :RDIVider[?]
      :RELock
      :SOURce[?]
      :SSCLock[?]
```

This subsystem has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:N1000:CRECovery:SOURce[?]</td>
<td>:N1000:CRECovery:ELEVel[?] on page 307</td>
</tr>
</tbody>
</table>
**:N1000:CRECovery:LSELect[?]**

**Syntax**

`:N1000:EMODules:SLOT:CONNect <identifier>, {ON | OFF | 1 | 0}`

`:N1000:CRECovery:LSELect[?]` on page 308

**Input Parameters**

- `identifier`: 'DCA*.SLOT*'
- ON | OFF | 1 | 0

**Return Range**

ON | OFF | 1 | 0

**Description**

This command enables usage of the addressed slot by M8070B. In case of using multiple M8070B instances, make sure to enable a slot only in exactly one M8070B instance.

This query returns the present state.

This SCPI is applicable for M8046A.

**Example**

`:N1000:EMOD:SLOT:CONN 'DCA1.SLOT1',ON`

**NOTE**

This command is not the same as the similar FlexDCA command

`:EMODules:SLOT{n}:CONNect[?] {ON | OFF | 1 | 0}`

The above command must be sent to FlexDCA prior starting up the M8070B software instances that shall control FlexDCA!

Managing the slot usage and connection state must be done using the FlexDCA commands or user interface before starting up M8070B.
:[:N1000:EMODules:SLOT:MODel?]
Syntax :[:N1000:EMODules:SLOT:MODel? <identifier>]
Parameters 'identifier': 'DCA*.SLOT*'
Description This query returns the model number of all DCA related modules which are connected to M8070B system software.
This SCPI is applicable for M8046A.
Example :[:N1000:EMOD:SLOT:MOD? 'DCA1.SLOT1']

[:N1000:CRECovery:CLBandwidth[?]]
Syntax :[:N1000:CRECovery:CLBandwidth <identifier>,<bandwidth>]
[:N1000:CRECovery:CLBandwidth? <identifier>]
Parameters 'identifier': 'DCA*.SLOT*'
Description This command sets the fix loop bandwidth.
This query returns the present state.
This SCPI is applicable for M8046A.
Example :[:N1000:CREC:CLB 'DCA1.SLOT1',10e6]
For detailed documentation, see :CRECovery{n}:CLBandwidth in the FlexDCA online help.

[:N1000:CRECovery:CRATe?]
Syntax :[:N1000:CRECovery:CRATe? <identifier>]
Parameters 'identifier': 'DCA*.SLOT*'
Description This query returns the clock recovery’s symbol rate.
This SCPI is applicable for M8046A.
Example :[:N1000:CREC:CRAT? 'DCA1.SLOT1']
For detailed documentation, see :CRECovery{n}:CRATe in the FlexDCA online help.
**:N1000:CRECoyer:ELEVel[?]**

**Syntax**: 
:N1000:CRECoyer:ELEVel <identifier>,<NRf>

:N1000:CRECoyer:ELEVel? <identifier>

**Input Parameters**: 
'identifier': 'DCA*.SLOT*'

**Description**: This command sets the equalizer at the electrical inputs of the clock recovery. The equalizer level range is 0 to 100. This query returns the current settings. This SCPI is applicable for N1076B and N1078A.

**Example**: 
:N1000:CREC:ELEVel 'DCA1.SLOT1',12

For detailed documentation, see :CRECoyer{n}:ELEVel in the FlexDCA online help.

**:N1000:CRECoyer:LBWMode[?]**

**Syntax**: 
:N1000:CRECoyer:LBWMode <identifier>,{FIXed | RDEPendent}

:N1000:CRECoyer:LBWMode? <identifier>

**Input Parameters**: 
'identifier': 'DCA*.SLOT*'

**Description**: This command sets the loop bandwidth mode to either fixed or symbol rate dependent. This query returns the present state. This SCPI is applicable for M8046A.

**Example**: 
:N1000:CREC:LBWM 'DCA1.SLOT1',RDEP

For detailed documentation, see :CRECoyer{n}:LBWMode in the FlexDCA online help.
:N1000:CRECovy:LOCKed?

Syntax :N1000:CRECovy:LOCKed? <identifier>

Input 'identifier': 'DCA*.SLOT*'

Parameters

Description This query displays the clock recovery's lock status.

Response: 1 (for locked) or 0 (for unlocked)

This SCPI is applicable for M8046A.

Example :N1000:CREC:LOCK? 'DCA1.SLOT1'

For detailed documentation, see :CRECovy(n):LOCKed the FlexDCA online help.

:N1000:CRECovy:LSELect[?]

Syntax :N1000:CRECovy:LSELect <identifier>, <LOOP1 | LOOP2 | LOOP3 | LOOP4>

:N1000:CRECovy:LSELect? <identifier>,

Input 'identifier': 'DCA*.SLOT*'

Parameters

Description This command selects the type 2 loop transition frequency.

This query returns the present state.

This SCPI is applicable for M8046A.

Example :N1000:CREC:LSEL 'DCA1.SLOT1', LOOP1

For detailed documentation, see :CRECovy(n):LSELect in the FlexDCA online help.

:N1000:CRECovy:LSELect:AUTomatic[?]

Syntax :N1000:CRECovy:LSELect:AUTomatic <identifier>, <ON | OFF | 1 | 0>

:N1000:CRECovy:LSELect:AUTomatic? <identifier>

Input 'identifier': 'DCA*.SLOT*'

Parameters ON | OFF | 1 | 0

Description This command enables or disables automatic selection of the type 2 loop
transition frequency.
This query returns the present state.
This SCPI is applicable for M8046A.

Example
:N1000:CREC:LSEL:AUTO 'DCA1.SLOT1',ON
For detailed documentation, see :CRECovery{n}:LSELect:AUTO in the FlexDCA online help.

:N1000:CREC:ODR?
Syntax :N1000:CREC:ODR? <identifier>
Input Parameters
Input Parameters 'identifier': 'DCA*.SLOT*' 
Return Range SUB32 | SUB16 | SUB8 | SUB4 | SUB2 | UNI티 | SUPer2 | SUPer4 | SUPer8
Description This query returns the recovered clock’s output divide ratio.
This SCPI is applicable for M8046A.
Example :N1000:CREC:ODR? 'DCA1.SLOT1'
For detailed documentation, see :CREC:ODR in the FlexDCA online help.

:N1000:CREC:ODR:AUTO?
Syntax :N1000:CREC:ODR:AUTO? <identifier>
InputParameters Input Parameters 'identifier': 'DCA*.SLOT*' 
Return Range ON | OFF | 1 | 0
Description This query returns the present state.
This SCPI is applicable for M8046A.
Example :N1000:CREC:ODR:AUTO? 'DCA1.SLOT1'
For detailed documentation, see :CREC:ODR:AUTO in the FlexDCA online help.
:MEAS:BER?
Syntax :MEAS:BER? [ival,]{FREQuency | PERTh}\nInput 'ival': '0..3'
Parameters
Description This query returns the Bit Error Rate.
Example :MEAS:BER? 1
For detailed documentation, see :MEASurement:BER in the FlexDCA online help.
Input Parameters

- 'identifier': 'DCA*.SLOT*'

Description

This command selects the input of the clock recovery.

This query returns the present state.

This SCPI is applicable for M8046A.

Example

```
:N1000:CREC:SOUR 'DCA1.SLOT1',DIFF
```

For detailed documentation, see :CREcovery(n):SOURce in the FlexDCA online help.

---

: N1000:CREcovery:SSCLock [?]

Syntax

```
:N1000:CREcovery:SSCLock [?] <identifier>,{OFF | ON | 0 | 1}
:N1000:CREcovery:SSCLock? <identifier>
```

Input Parameters

- 'identifier': 'DCA*.SLOT*'

Description

This command turns on the ability to use the nominal symbol rate when recovering the clock on Spread Spectrum Clock (SSC) signals. Normally, FlexDCA uses the average symbol rate to recover the clock. However, when performing clock recovery on a SSC signal, using the nominal symbol rate may be required. The default setting is off.

This command requires FlexDCA revision A.05.80 and above, and is only available for clock recovery modules with Option JSA (see the Electrical and Optical Clock Data Recovery Solutions data sheet, 5992-1620EN)

This query returns the present state.

This SCPI is applicable for M8046A.

Example

```
:N1000:CREC:SSCL 'DCA1.SLOT1',ON
```

For detailed documentation, see :CREcovery(n):SSCLock in the FlexDCA online help.
FETCh Subsystem

The FETCh subsystem is used to query the error detector’s results. Commands for starting, stopping, resetting, etc. are found in the :PLUGin:ERATio subsystem.

This subsystem has the following SCPI structure:

```
:FETCh
"BCOunter?
"IBERate?
```

This subsystem has the following commands:

**Table 67**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:BCOunter?</td>
<td>:FETCh:BCOunter? on page 312</td>
</tr>
<tr>
<td>:IBERate?</td>
<td>:FETCh:IBERate? on page 315</td>
</tr>
</tbody>
</table>

**:FETCh:BCOunter?**

**Syntax**: 
:FETCh:BCOunter? 'identifier'

**Input Parameters**:
'identifier': 'M*.DataIn1', 'M*.DataIn2', or 'M*.DataIn', 'INF*.DataIn1'

**Description**: 
This query returns the accumulated bit counts and coding specific counts since start of accumulation period. The results are combined to an expression. An expression is comparable to a struct in a programming language. It’s a set of parameter of different types separated by a comma and enclosed by parenthesis.

The returned expression contains a timestamp, the location and a set of bit counter values.

(TimeStamp, “Location”, Counted1s, Counted0s, Erroneous1s, Erroneous0s[, coding specific counters])
**Timestamp:** As timestamp the UNIX epoch time is used; it is defined as elapsed seconds, milliseconds etc. since 01.01.1970 UTC.

**Location:** It’s a string referred to the place the bit counter are located. This parameter is important if a group name is used as ‘Identifier’.

**Counted1s:** The number of received ones since start of accumulation period

**Counted0s:** The number of received zeros since start of accumulation period.

**Erroneous1s:** The number of received erroneous ones since start of accumulation period.

**Erroneous0s:** The number of received erroneous zeros since start of accumulation period.

**Additional Counters for 8b10b coding:**
- ComparedSymbols
- ErroredSymbols
- IllegalSymbols
- WrongDisparity
- FillerSymbols
- ReceivedSymbols
- Frames
- ErroredFrames

**Additional Counters for 128/130 and 128/132 coding:**
- Blocks
- ErroredBlocks
- IllegalSyncHeaders
- FillerSymbols
- ModifiedFillerSymbols
- Frames
- ErroredFrames
- Additional Counter for 128/132 coding
- CorrectedSyncHeaders
Additional Counters for PAM4 line coding
- Compared Symbols
- Errored Symbols
- Compared Symbol0
- Errored Symbol0
- Compared Symbol1
- Errored Symbol1
- Compared Symbol2
- Errored Symbol2
- Compared Symbol3
- Errored Symbol3

If this query is sent to a ‘group’ of inputs, the response will be expanded by additional sets of counted values. The expression contains for every input belonging to a group a separate set of values starting with the location.

This query returns the present state.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8046A and real-time oscilloscope based error detector channels.

Example :FETCh:BCO? 'M1.DataIn1'
Response: (1.366014336e9,'M1.DataIn1',1.2222e10,1.112345e10,5.1e1,7.2e2)
:FETCh:BCO? 'Lanes'
Response: (1.366014336e9,'M1.DataIn1',1.2222e10,1.112345e10,5.1e1,7.2e2, 'M1.DataIn2', ...,...)

NOTE
Unix time or POSIX time, is a system for describing instances in time, defined as the number of seconds that have elapsed since midnight Coordinated Universal Time (UTC), 1. January 1970, not including leap seconds.
:FETCH:IBERate?

Syntax: :FETCH:IBERate? 'identifier'

Input Parameters:

- 'identifier': 'M*.DataIn1' or 'M*.DataIn2', 'INF*.DataIn1'

Description: This query returns the instantaneous bit error rate measured by the addressed error detector. The response contains a timestamp, the location and the measured BER.

(TimeStamp, "Location", BER[, coding specific ratios])

- **TimeStamp**: As timestamp the Unix epoch time is used; it is defined as elapsed seconds, milliseconds etc., since 01.01.1970 UTC.
- **Location**: It is a string referred to the place the ratios are located. This parameter is important if a group name is used as 'Identifier'.
- **BER**: The instantaneous bit error ratio.

Additional counters for 8b10b coding:

- SymbolErrorRatio
- IllegalSymbolRatio
- DisparityErrorRatio
- FillerSymbolRatio
- FrameErrorRatio

Additional Ratios for 128/130 and 128/132 coding:

- BlockErrorRatio
- FillerSymbolRatio
- FrameErrorRatio

Additional Counters for PAM4 line coding:

- Symbol Error Ratio
- Symbol0 Error Ratio
- Symbol1 Error Ratio
- Symbol2 Error Ratio
- Symbol3 Error Ratio

This query returns the present state.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8046A and real-time oscilloscope based error detector channels.
As 'Identifier' a group name is also allowed. The response will be expanded by additional sets of values per error detector included in the group.

Example

: FETC:IBER? 'M6.DataIn1'
Response: (1.266014336e9, 'M6.DataIn1', 1.2345e-1)

: FETC:IBER? 'Lanes'
Response: (1.266514336e9, 'M1.DataIn1', 1.2345e-1, 'M1.DataIn2', 0.0e0)
MMEMory Subsystem

All instrument states of the M8020A/M8030A instrument are stored in a "workspace". There are two defined areas in a workspace. One area is named "Factory". In this area you can find read only predefined settings and patterns. The other area is named "User". This area stores customer created settings or patterns.

The "Path" argument is a simple string that can be used to address a setting. This is a simple name that follows the Windows specification for allowed characters of a file name. Additionally, group or folder names can be specified for grouping settings. For example, "SATA/Test1" or "SATA/InitialTests/TestAtTemperature80" and "SATA/InitialTests/TestAtTemperature32" or "PCIeStartUpParameter".

This subsystem has the following SCPI structure:

```
+MMEMory
  +WORKspace
    +SETting
      +USER
        +CATalog?
        +STORe
        +RECall
        +DELETE
      +FACTory
        +CATalog?
        +RECall
      +IMPort
        +[USER]
          +CATalog?
          +DELETE
```
This subsystem has the following commands:

Table 68

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:WORKspace:SETTINGs:USER:STORe</td>
<td>:MMEMory:WORKspace:SETTINGs[:USER]:STORe on page 319</td>
</tr>
<tr>
<td>:WORKspace:SETTINGs:USER:RECAll</td>
<td>:MMEMory:WORKspace:SETTINGs[:USER]:RECAll on page 319</td>
</tr>
<tr>
<td>:WORKspace:SETTINGs:USER:DELele</td>
<td>:MMEMory:WORKspace:SETTINGs[:USER]:DELele on page 320</td>
</tr>
<tr>
<td>:WORKspace:SETTINGs:FACTory:RECAll</td>
<td>:MMEMory:WORKspace:SETTINGs:FACTory:RECAll on page 320</td>
</tr>
<tr>
<td>:WORKspace:IMPort[:USER]:DELele</td>
<td>:MMEMory:WORKspace:IMPort[:USER]:DELele on page 322</td>
</tr>
</tbody>
</table>
:MMEMory:WORKspace:SETTings[:USER]:CATalog?

Syntax :MMEMory:WORKspace:SETTings[:USER]:CATalog?

Description This query returns a list of names currently stored in the 'User' area.

Example :MMEM:WORK:SETT:CAT?

'PCI3_InitialTestSetting, DefaultSetting_1, DefaultSetting_2, DefaultSetting_3, DefaultSetting_4, PCI3/InitialTestA, PCI3/InitialTestB, PCI3_InitialTestSetting'

:MMEMory:WORKspace:SETTings[:USER]:STORe

Syntax :MMEMory:WORKspace:SETTings[:USER]:STORe 'Path'

Input Parameters

'Path': Specify name/path to store instrument state.

Description This command stores the instrument state to the specified location. The 'Path' argument of this command can be a simple string like 'TestConfigurationPCIe3'; the instrument state will then be stored into a location named 'TestConfigurationPCIe3'. In addition, it can specify a group or folder name. This command is meant for grouping settings into one folder (for example, 'PCIe1/TestConfiguration1').

Example :MMEM:WORK:SETT:STOR 'TestConfigurationSATA_1'
:MMEM:WORK:SETT:STOR 'SATA/TestConfiguration_1'
:MMEM:WORK:SETT:STOR 'SATA/TestConfiguration_2'

:MMEMory:WORKspace:SETTings[:USER]:RECall

Syntax :MMEMory:WORKspace:SETTings[:USER]:RECall 'Path'

Input Parameters

'Path': Specify location of stored instrument state.

Description This command recalls/restores the instrument state from the specified location.

Example :MMEM:WORK:SETT:REC 'TestConfigurationSATA_1'
:MMEM:WORK:SETT:REC 'SATA/TestConfiguration_1'
:MMEM:WORK:SETT:REC 'SATA/TestConfiguration_2'
:**MMEMory:WORKspace:SETTings[:USER]:DELete**

**Syntax**  
:MMEMory:WORKspace:SETTings[:USER]:DELete 'Path'

**Input**  
'Path': Specify location of stored instrument state.

**Parameters**

**Description**  
This command deletes the instrument state from the specified location.

**Example**  
:MMEM:WORK:SETT:DEL 'TestConfigurationSATA_1'
:MMEM:WORK:SETT:DEL 'SATA/TestConfiguration_1'
:MMEM:WORK:SETT:DEL 'SATA/TestConfiguration_2'

:**MMEMory:WORKspace:SETTings:FACTory:CATalog?**

**Syntax**  
:MMEMory:WORKspace:SETTings:FACTory:CATalog?

**Description**  
This command creates a list of “predefined” setting names located in the “Factory” area of the workspace.

**Example**  
:MMEM:WORK:SETT:FACT:CAT? 'PCIe3_InitialTestSetting'

:**MMEMory:WORKspace:SETTings:FACTory:RECall**

**Syntax**  
:MMEMory:WORKspace:SETTings:FACTory:RECall 'Path'

**Input**  
'Path': Specify location of stored instrument state.

**Parameters**

**Description**  
This command recalls/restores the instrument state from a “factory predefined” addressed location.

**Example**  
:MMEMory:WORKspace:SETTings:FACTory:RECall 'PCIe1/PCIe1_compliance_ASIC'
:MMEMory:WORKspace:IMPort[:USER]

Syntax: :MMEMory:WORKspace:IMPort[:USER]

.ProfileType>, <"SourcePath">, <"RelativeTargetPath">

Input Parameters:

(ProfileType): Specify jitter sweep or spread spectrum clocking profile type or SPProfile.

("SourcePath"): Specify the absolute source path of file to be imported.

("RelativeTargetPath"): Specify a current setting or a shared (common) storage location.

Description: An arbitrary profile waveform file can be imported for generating jitter sweep or spread spectrum clocking. Three arguments have to be specified:

The first one is the profile type JSPProfile | SSCProfile | SPProfile,
second one is the absolute source path of the file to be imported,
third one is a relative target path "current:..." addresses the current setting and "shared: ..." addresses a location in the storage that is common for all user settings.

For jitter sweep files, the file extension "...jcs" has to be used and for spread spectrum clocking the extension "...txt".

S-parameter profiles containing 1- to 9-port components are supported. The file extension ".s1p...s9p" can be used.

Example: :MMEM:WORK:IMP
jspr,"c:/temp/Jittersweep.jcs","shared:Jittersweep.jcs"

Example: :MMEM:WORK:IMP
jspr,"c:/temp/Jittersweep.jcs","current:Jittersweep.jcs"

Example: :MMEM:WORK:IMP
jspr,"c:/temp/Jittersweep.jcs","current:test/PCIe3.0/Jittersweep.jcs"

Example: :MMEM:WORK:IMP
ssp,"c:/temp/INCH7_7_88.s4p","shared:INCH7_7_88.s4p"

Example: :MMEM:WORK:SSC
ssc,"c:/temp/INCH7_7_88.s4p","shared:INCH7_7_88.s4p"
:MMEMory:WORKspace:IMP[:USER]:CATalog?

Syntax :MMEMory:WORKspace:IMP[:USER]:CATalog? <ProfileType>

Input <ProfileType>: Specify jitter sweep, spread spectrum clocking and spread spectrum profile type i.e. <JSPRofile | SSCProfile | SPProfile>.

Description This query generates a list of imported files based on the profile type.


Returns:
"current:Jittersweep.jcs", "shared:Jittersweep.jcs", "shared:Sweep.jcs", "shared:Test42.jcs"

:MMEMory:WORKspace:IMP[:USER]:DELete

Syntax :MMEMory:WORKspace:IMP[:USER]:DELete <ProfileType>, <"RelativeTargetPath">

<ProfileType>: Specify jitter sweep, spread spectrum clocking and spread spectrum profile type i.e. <JSPRofile | SSCProfile | SPProfile>.

<"RelativeTargetPath">: Specify a current setting or a shared (common) storage location.

Description Deletes an imported profile waveform file from the specified storage location. First argument is the profile type and the second argument is the relative target path of the file to be deleted.

ACQuisition subsystem

The acquisition subsystem controls how a real-time oscilloscope is being configured in order to capture the signal for BER measurements. This is comparable to setting up the acquisition memory of the oscilloscope, but in terms of bits or symbols, and not in analog samples.

The channel bandwidth limit must be equal or less to the global bandwidth limit setting. The available bandwidth limit depends on the oscilloscope model being used.

Additionally, the acquisition can be stopped in M8070B, so that the oscilloscope becomes free for manual adjustments, while still being under parametric control of M8070B.

This subsystem has the following SCPI structure:

```
:ACQuisition
  .:BANDwidth
    .:MODE[?]
    .:STATe[?]
    .:TYPE[?]
    .:VALue[?]
  .:BITS[?]
  .:POINts[:ANALog]?
    .:REConfigure
    .:STATe[?]
    .:SYMBols[?]
```
This subsystem has the following commands:

Name | Description under
---|---

### :ACQuisition:BANDwidth:MODE?

**Syntax**

:ACQuisition:BANDwidth:MODE <identifier>,{AUTO | MANual | MAXimum}

:ACQuisition:BANDwidth:MODE? <identifier>

**Identifier**

`INF*.Common`

**Description**

This command controls the global bandwidth limit setting of the real-time oscilloscope. The available bandwidth limit depends on the oscilloscope model being used.

This query returns the present setting.

**Example**

:ACQuisition:BANDwidth:STATe[?] 

Syntax :ACQuision:BANDwidth:STATe <identifier>,{OFF | ON | 0 | 1} 

Identifier 'INF*.DataIn1'

Description This command enables or disabled the channel bandwidth limit of the real-time oscilloscope. The channel bandwidth limit must be equal or less to the global bandwidth limit setting.

The available bandwidth limit depends on the oscilloscope model being used.

This query returns the present setting.

Example :ACQ:BAND:STAT 'INF1.DataIn1',ON

:ACQuisition:BANDwidth:TYPE[?] 

Syntax :ACQuision:BANDwidth:TYPE <identifier>,{BESSEL4 | WALL} 

Identifier 'INF*.DataIn1'

Description This command controls the channel bandwidth limit filter of the real-time oscilloscope.

The available bandwidth limit depends on the oscilloscope model being used.

This query returns the present setting.

Example :ACQ:BAND:TYPE 'INF1.DataIn1',BESSEL4

:ACQuisition:BANDwidth:VALue[?] 

Syntax :ACQuision:BANDwidth:VALue <identifier>,<bandwidth> 

Identifier 'INF*.Common' or 'INF*.DataIn1'

Description This command controls the bandwidth limit of the real-time oscilloscope.

The available bandwidth depends on the oscilloscope model being used.
This query returns the present setting.

Example :ACQ:BAND:VAL 'INF*.Common',40e9

:ACQuisition:BITS[?]

Syntax :ACQuisition:BITS<identifier>,<number_of_bits>

:ACQuisition:BITS? <identifier>

Identifier 'INF*.DataIn1'

Description This command configures the number of bits to capture and process per acquisition. This does directly control the smallest detectable BER value per acquisition, which corresponds to the instantaneous BER result (see :FETCH:IBERate? on page 315).

Increasing the number of bits per acquisition will lower the detectable BER, but at the same time increase the processing duration per acquisition.

This query returns the present setting.

Example :ACQ:BITS 'INF1.DataIn1',1e6

:ACQuisition:POINts[:ANALog]?

Syntax :ACQuisition:POINts[:ANALog]? <identifier>

Identifier 'INF*.DataIn1', 'INF*.Common'

Description This query returns the number of analog sample points that are configured into the oscilloscope. This value is for information only, and derives from the configured number of bits per acquisitions and the channel’s symbol rate.

Example :ACQ:POIN? 'INF1.DataIn1'

Returns 2.96548e6

:ACQuisition:POINts:REConfigure

Syntax :ACQuisition:POINts:REConfigure <identifier>

Identifier 'INF*.DataIn1'
Description  This command performs a reset and full reprogramming of the oscilloscope. It effectively re-applies the M8070B settings and resets the manual changes.

Example  :ACQ:POIN:REC 'INF1.DataIn1'

:ACQuisition:STATe[?]

Syntax  :ACQuisition:STATe <identifier>, {OFF | ON | 0 | 1}

:ACQuisition:STATe? <identifier>

Identifier  'INF*.Common'

Description  This command enables or disables the acquisition. If <identifier> addresses a location that is not a channel, then the acquisition on all channels of the oscilloscope will be controlled by this command.

This query returns the present setting.

Example  :ACQ:STAT 'INF1.Common',OFF

:ACQuisition:SYMBols[?]

Syntax  :ACQuisition:SYMBols <identifier>,<number_of_UI>

:ACQuisition:SYMBols? <identifier>

Identifier  'INF*.DataIn1'

Description  This command configures the number of symbols to capture and process per acquisition. This does directly control the smallest detectable BER value per acquisition, which corresponds to the instantaneous BER result (see :FETCh:IBERate? on page 315).

Increasing the number of symbols per acquisition will lower the detectable BER, but at the same time increase the processing duration per acquisition. In case of NRZ, this command will produce the same results as :ACQ:BITS (see :ACQuisition:BITS[?] on page 326), in case of PAM4 is will configure the number of bits to twice the number of symbols.

This query returns the present setting.

Example  :ACQ:SYMB 'INF1.DataIn1',1e6
PLUGin Subsystem

The M8020A/M8030A/M8040A platform supports a plugin interface used to implement certain interfaces recognized by the software and integrated into the M8070B GUI and instrument software. Error Ratio, Output Timing, Jitter Tolerance, Output Level, Eye Diagram, Parameter Sweep, Pattern Capture and Error Distribution Analysis measurements are examples of the plugin concept.

For more information about the plugin interface, refer to PLUGin Subsystem on page 23.

Out of these measurements, Error Ratio measurement and Pattern Capture is available with M8070B software system. However, all other measurements are the part of plugins and will only be available on installing them in M8070B software system. The following plugins are available:

- **Advanced Measurement Package** - The Advanced Measurement Package provides the following measurements:
  - Output Timing
  - Output Level
  - Jitter Tolerance
  - Eye Diagram
  - Parameter Sweep

  For details on the measurements provided by Advanced Measurement Package, please refer to *M8000 Series Advance Measurement Package User Guide*.

- **Error Distribution Analysis Package** - The Error Distribution Analysis Package provides the following measurement:
  - Error Distribution Analysis

  For details on the measurement provided by Error Distribution Analysis Package, please refer to *M8000 Series Error Distribution Analysis Package User Guide*.

**NOTE**

The Advanced Measurement Package and Error Distribution Analysis Package requires a valid license for activation.
This subsystem has the following SCPI structure:

```plaintext
:PLUGin
  :CATalog?
  :RNODEs
  :LIST?

  Default measurements provided by M8070B System Software

  :ERATio
  ...
  :CCAPture
  ...

  Additional measurements provided by M8070B System Software on installing the Advance Measurement Package

  :OTIMing
  ...
  :OLEVel
  ...
  :JTOLerance
  ...
  :EDIagram
  ...
  :PSWEsp
  ...

  Additional measurement provided by M8070B System Software on installing the Error Distribution Analysis Package

  :EDA
  ...
```
This subsystem has the following subnodes:

Table 70

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ERATio Subnode</td>
<td>:PLUGin:ERATio Subnode on page 331</td>
</tr>
<tr>
<td>:CCAPture Subnode</td>
<td>:PLUGin:CCAPture Subnode on page 352</td>
</tr>
<tr>
<td>Advanced Measurement Package</td>
<td>Advanced Measurement Package on page 371</td>
</tr>
<tr>
<td>Error Distribution Analysis Package</td>
<td>Error Distribution Analysis Package on page 371</td>
</tr>
</tbody>
</table>

:PLUGin:CATalog?

**Syntax**
:PLUGin:CATalog?

**Description**
This query returns the names of all the plug-ins.

**Example**
:PLUGin:CATalog?


:PLUGin:RNODes[:LIST]?

**Syntax**
:PLUGin:RNODes[:LIST]?

**Description**
This query returns a list of SCPI plugin root nodes that can be used to address different type of measurements (plugins).

**Example**
:PLUG:RNOD:LIST?

"CCAPture", "ERATio", "JTOlerance", "OLEVel", "OTIMing", "LTXGKR", "EDlag ram", "PSWEep", "OCEI", "CPHYplugin"
This is the basic measurement for the bit error ratio, symbol error ratio and frame error ratio and is integrated with the M8070B software during the startup. The measurement comes with the set of acquisition and evaluation parameters which are used to configure the measurement run. The acquisition parameters can only be configured before the measurement starts (i.e., you cannot change the acquisition parameter number while the measurement is running); however, the evaluation parameters can be configured at any time, even while the measurement is running.

Results can be retrieved using the :PLUGin:ERATio:FETCh:DATA? or :FETCh subsystem commands.

The M8040A system also supports error ratio measurement using NRZ or PAM4.
This subnode has the following SCPI structure:

```
.:PLUGin
  .:ERATio
    .:ACQuisition
    .:ALOCation[?]
    .:AEND[?]
    .:COMPared[?]
    .:DURation[?]
    .:HISTory[?]
    .:INTerval[?]
    .:TCLevel[?]
    .:TERatio[?]
    .:TIME[?]
    .:EVALuation
    .:CERatio[?]
    .:RVMode?
    -:FETCh
    .:.DATA?
    [:DURation]?
    .:.PFResult?
    .:CATalog?
    .:DELete
    .:NEW
    .:RESet
    .:STARt
    .:STOP
    .:SHOW:SGLegends[?]
    .:SHOW:COMMent[?]
    .:GRAPh?
    .:RUN
    .:HISTory
      [:STATe][?]
      .:CLEar
      .:LOG?
      .:MESSage?
      .:PROGress?
      [:STATus]?
```
This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DELete</td>
<td>:PLUGin:ERATio:DELete on page 345</td>
</tr>
<tr>
<td>:NEW</td>
<td>:PLUGin:ERATio:NEW on page 345</td>
</tr>
<tr>
<td>:RESET</td>
<td>:PLUGin:ERATio:RESet on page 346</td>
</tr>
<tr>
<td>:START</td>
<td>:PLUGin:ERATio:START on page 346</td>
</tr>
<tr>
<td>:STOP</td>
<td>:PLUGin:ERATio:STOP on page 347</td>
</tr>
<tr>
<td>Name</td>
<td>Description under</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>:RUN:[STATus]?</td>
<td>:PLUGin:ERATio:RUN:[STATus]? on page 351</td>
</tr>
</tbody>
</table>
:PLUGin:ERATio:ACQuisition:ALOCation[?]
Syntax  :PLUGin:ERATio:ACQuisition:ALOCation 'identifier', <location-string>
         :PLUGin:ERATio:ACQuisition:ALOCation? 'identifier'
Input   'identifier': Specify the measurement name.
Parameters <location-string>: Specify the location identifier or group name identifier.
Return Range <location or group name>
Description This command sets the location identifier (or group name identifier) against which the data acquisition is performed for the specified measurement name identifier. The query returns the current location or group name for the specified measurement name.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.
Example  :PLUG:ERAT:ACQ:ALOC 'MyMeasurement','M1.DataIn1'

:PLUGin:ERATio:ACQuisition:AEND[?]
Input   'identifier': Specify the measurement name.
Parameters <PFA|FDUR|CBIT>: Set accumulation end mode.
Return Range PFA|FDUR|CBIT
Description This command sets the accumulation to end when a pass/fail (PFA) condition has been met or end after the full duration (FDUR) or no.of bits (CBIT) of the measurement for the addressed measurement name identifier. The query returns the current accumulation end value.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.
Example  :PLUG:ERAT:ACQ:AEND 'MyMeasurement', PFA
:PLUGin:ERATio:ACQuisition:SCRiteria[?]


Input     'identifier': Specify the measurement name.
Parameters <PAN|CAN>: Set accumulation stop criteria.

Return Range    PAN|CAN

Description     This command sets the accumulation stop criteria as either "Per Analyzer" or "Combined Analyzer". The combined result will appear in form of Graph, Data Tab and Calculated Results.

• Per Analyzer (PAN) - Graph, Data Tab and combined Calculated Results will not be visible.
• Combined Analyzer (CAN) - Graph, Data Tab and combined Calculated Results will be visible.

NOTE

This command is only valid when a group is selected as an analyzer (If the group is having only one location, this command is not applicable).

This command depends on "Accumulation End". It will not be not applicable if the "Accumulation End" is set as "Full Duration".

The query returns the current accumulation end value.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example    :PLUG:ERAT:ACQ:SCR 'MyMeasurement', CAN

:PLUGin:ERATio:ACQuisition:COMPared[?]

Syntax    :PLUGin:ERATio:ACQuisition:COMPared 'identifier', <NRf>
       :PLUGin:ERATio:ACQuisition:COMPared? 'identifier'

Input     'identifier': Specify the measurement name.
Parameters <NRf>: IE+6 to IE+18

Return Range    IE+6 to IE+18
Description
This command sets the Number of Compared Bits for the addressed measurement name identifier. The query returns the current value.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example
:PLUG:ERAT:ACQ:COMP 'MyMeasurement', IE+8

:PLUGin:ERATio:ACQuisition:DURation[?]

Syntax
:PLUGin:ERATio:ACQuisition:DURation 'identifier', <FTIM|IND>

Input
'identifier': Specify the measurement name.

Parameters
<FTIM|IND>: Set accumulation duration mode.

Return Range
FTIM|IND

Description
This command sets the accumulation duration mode to either a fixed time (FTIM) or indefinitely (IND) for the addressed measurement name identifier. If FTIM is selected, use the :PLUGin:ERATio:ACQuisition:TIME command to set the duration time. The query returns the current accumulation duration value.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example
:PLUG:ERAT:ACQ:DUR 'MyMeasurement', FTIM

:PLUGin:ERATio:ACQuisition:HISTory[?]

Syntax
:PLUGin:ERATio:ACQuisition:HISTory 'identifier', <NRf>

Input
'identifier': Specify the measurement name.

Parameters
<NRf>: Set size of accumulation values.

Return Range
1 to 100000

Description
This command sets the size of accumulation values to be kept in memory for the addressed measurement name identifier. The query returns the current history size.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.
Example :PLUG:ERAT:ACQ:HIST 'MyMeasurement', 100000

:PLUGin:ERATio:ACQuisition:INTerval

Syntax :PLUGin:ERATio:ACQuisition:INTerval 'identifier', <NRf>


Input 'identifier': Specify the measurement name.
Parameters <NRf>: Set accumulation interval.

Return Range 100E-3 to 2E6

Description This command sets how often to take error ratio samples (from 0.1 to 2000000 seconds) for the addressed measurement name identifier. The query returns the current accumulation interval.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

NOTE Do not use the "Ms" unit with a value. Doing so will cause an error. Instead, use "mas" for the unit. For example, 2E6 is the same as 2 mas.

Example :PLUG:ERAT:ACQ:INT 'MyMeasurement', 0.3
### :PLUGin:ERATio:ACQuisition:TCLevel\[?\]

**Syntax**

```
:PLUGin:ERATio:ACQuisition:TCLevel 'identifier', <NRf>
:PLUGin:ERATio:ACQuisition:TCLevel? 'identifier'
```

**Input Parameters**

- `'identifier'`: Specify the measurement name.
- `<NRf>`: Set target confidence level.

**Return Range**

0.1% to 99.9%

**Description**

This command sets the target confidence level when accumulation end is set to pass/fail (in percent) for the addressed measurement name `identifier`. The query returns the current target confidence level.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

**Example**

```
:PLUG:ERAT:ACQ:TCL 'MyMeasurement', 95
```

### :PLUGin:ERATio:ACQuisition:TERatio\[?\]

**Syntax**

```
:PLUGin:ERATio:ACQuisition:TERatio 'identifier', <NRf>
:PLUGin:ERATio:ACQuisition:TERatio? 'identifier'
```

**Input Parameters**

- `'identifier'`: Specify the measurement name.
- `<NRf>`: Set target error ratio.

**Return Range**

1E-18 to 1E-3

**Description**

This command sets the target error ratio for the addressed measurement name `identifier`. The query returns the current target error ratio value.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

**Example**

```
:PLUG:ERAT:ACQ:TER 'MyMeasurement', 1e-09
```
:PLUGin:ERATio:ACQuisition:TIME[?]

Syntax :PLUGin:ERATio:ACQuisition:TIME 'identifier', <NRf>

Input 'identifier': Specify the measurement name.
Parameters <NRf>: Set accumulation fixed time.

Return Range 1 s to 31.5E6 s

Description This command sets the accumulation fixed time, when accumulation duration is set to FTIM (fixed time), for the addressed measurement name identifier. The query returns the current accumulation fixed time value in seconds.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example :PLUG:ERAT:ACQ:TIME 'MyMeasurement',1000

NOTE Do not use the “Ms” unit with a value. Doing so will cause an error.
Instead, use “mas” for the unit. For example, 20E6 is the same as 20 mas.

:PLUGin:ERATio:SHOW:CERatio[?]

Syntax :PLUGin:ERATio:SHOW:CERatio 'identifier', <ERAT|EZR|EOR>
 :PLUGin:ERATio:SHOW:CERatio? 'identifier'

Input 'identifier': Specify the measurement name.
Parameters <ERAT|EZR|EOR>: Set error ratio analysis mode.

Return Range ERAT|EZR|EOR

Description This command sets the error ratio analysis mode to error ratio of all accumulated bits (ERAT), errored zero and counted zero bits (EZR), or errored ones and counted ones bits (EOR) for the addressed measurement name identifier. The query returns the current setting error ratio analysis mode.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example :PLUG:ERAT:SHOW:CER 'MyMeasurement',EOR
:PLUGin:ERATio:SHOW:RVMode[?]  
**Syntax**: :PLUGin:ERATio:SHOW:RVMode 'identifier', <DET|SUMM>  


**Input Parameters**  
'identifier': Specify the measurement name.  

<DET|SUMM>: Select measurement results view mode.  

**Return Range**  
DET|SUMM  

**Description**  
This command sets the measurement results view mode to detailed or summary when viewing the results using the :PLUGin:ERATio:FETCH:DATA? command. This mode can be changed during a measurement or after a measurement has completed. The default is summary mode.  

DET (detailed) sets the mode to return the error ratio, compared bits, errored bits, errored zero ratio, compared zeroes, errored zeroes, errored one ratio, compared ones, errored ones, etc.  

SUMM (summary) sets the mode to return the error ratio, compared bits, errored bits, etc.  

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.  

**Example**  
:PLUG:ERAT:SHOW:RVM 'MyMeasurement', DET  

:PLUGin:ERATio:FETCh:DATA?  
**Syntax**: :PLUGin:ERATio:FETCh:DATA? 'identifier'[,'Location']  

**Input Parameters**  
'identifier': Specify the measurement name.  

[,'Location']: 'M*.DATAOUT1' or 'M*.DataOut2' (optional).  

**Return Range**  
'M*.DATAOUT1' or 'M*.DataOut2' (data results)  

**Description**  
This command returns the raw data of the error ratio measurement for the addressed measurement name identifier and location identifier. A measurement run on location identifier 'M1.DataIn1', for example, would return results specific to 'M1.DataIn1' only. If the location identifier is omitted and measurement data exists from multiple locations, results will be returned for every location.
The first item in the comma separated list is the location identifier name (for example, 'M1.DataIn1'). The successive values are dependent on the coding, shown in the order below and are repeated in the same order for subsequent measured points.

**Bit Error Counter**

TimeStamp, ComparedOnes, ComparedZeroes, ErroredOnes, ErroredZeroes, ...

**Counter for 8b/10b Bit Coding**

TimeStamp, ComparedOnes, ComparedZeroes, ErroredOnes, ErroredZeroes, ComparedSymbols, ErroredSymbols, WrongDisparity, FillerSymbols, ReceivedSymbols, Frames, ErroredFrames, ...

**Counter for 128/130 Bit Coding**

TimeStamp, ComparedOnes, ComparedZeroes, ErroredOnes, ErroredZeroes, Blocks, ErroredBlocks, IllegalSynHeaders, FillerSymbols, ModifiedFillerSymbols, Frames, ErroredFrames, ...

**Counter for 128/132 Bit Coding**

TimeStamp, ComparedOnes, ComparedZeroes, ErroredOnes, ErroredZeroes, Blocks, ErroredBlocks, IllegalSynHeaders, CorrectedSyncHeaders, FillerSymbols, ModifiedFillerSymbols, Frames, ErroredFrames, ...

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

**Example**

The following example shows the results returned for 128/130 Bit Coding.

```
:PLUG:ERAT:FETC:DATA? 'Error Ratio 1', 'M1.DataIn1'
```

Return:

```"M1.DataIn1",1420795894.76206,499999811,500000189,0,0,7692307,0,0,0,0,0, ...
```

If the location is omitted from the above example e.g. if the query is: `:PLUG:ERAT:FETC:DATA? 'Error Ratio 1', 'Error Ratio 1'` then the query returns the comma separated list of all the analyzer location group participating in the measurement.

Return:

```"M1.DataIn1",1420795894.76206,499999811,500000189,0,0,7692307,0,0,0,0,0, ...,"
```

```"M1.DataIn2",1420795894.76206,499999811,500000189,0,0,7692307,0,0,0,0,0, ...,"
```
The following example shows the results returned when line coding is selected as PAM4:

("M2.DataIn",1511414904.34319,6064099210500,6064099210500,0,0,60 64099210496,0,1516024802624,0,1516024802624,0,1516024802624,0,1 516024802624,0,...)

To get the measurement results in detailed or summary mode, refer to the :PLUGin:ERATio:SHOW:RVMode[?] on page 341.

:PLUGin:ERATio:FETCh[:DUration]?

Syntax :PLUGin:ERATio:FETCh[:DUration]? 'identifier'[,'Location']

Parameters

'identifier': Specify the measurement name.

[,'Location']: Specify the location identifier (optional).

Return Range <NR3>

Description This command returns the bit error ratio duration, nOneBits, nZeroBits, nOneBitErrs and nZeroBitErrs of the specified measurement name and location identifier. A measurement run on location 'M1.DataIn1', for example, would return results specific to 'M1.DataIn1' only. If the location identifier is omitted and measurement data exists from multiple locations, results will be returned for every location.

The following example shows the results returned when line coding is selected as PAM4:

("M2.DataIn",1511414904.34319,6064099210500,6064099210500,0,0,60 64099210496,0,1516024802624,0,1516024802624,0,1516024802624,0,1 516024802624,0,...)

To get the measurement results in detailed or summary mode, refer to the :PLUGin:ERATio:SHOW:RVMode[?] on page 341.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example :PLUG:ERAT:FETC? 'MyMeasurement','M1.DataIn1'
:PLUGin:ERATio:FETCh[:DURation]:PFResult?

Syntax      :PLUGin:ERATio:FETCh[:DURation]:PFResult? 'identifier', ['Location']

Input       'identifier': Specify the measurement name.

Parameters   ['Location']: Specify the location identifier (optional).

Return Range <NR3>

Description This command returns the results from the :PLUGin:ERATio:FETCh[:DURation]? query as well as four additional values of the specified measurement name and location identifier:

- ConfidenceLevelAtTargetErrorRatio
- MinimumErrorRatioAtTargetConfidenceLevel
- MaximumErrorRatioAtTargetConfidenceLevel
- PassFailedResult ('UNKNOWN', 'PASS', or 'FAIL' if the AEND parameter is set to 'PassFail')

If the AEND parameter is set to 'FullDuration', the result is the same as :PLUGin:ERATio:FETCh[:DURation]? except that the four additional values are set to “Unknown” (not applicable).

A measurement run on location 'M1.DataIn1', for example, would return results specific to 'M1.DataIn1' only. If the location identifier is omitted and measurement data exists from multiple locations, results will be returned for every location.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example :PLUG:ERAT:FETC:PFR? 'MyMeasurement','M1.DataIn1'

:PLUGin:ERATio:CATalog?

Syntax      :PLUGin:ERATio:CATalog?

Description This command returns a list of all created error ratio measurement names currently available for measuring.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.
Example Assume the following is a list of created error ratio measurement names:
:PLUG:ERAT:NEW 'ERAT_1'
:PLUG:ERAT:NEW 'ERAT_2'
:PLUG:ERAT:NEW 'ERAT_3'
The command and returned list would look like the following:
:PLUG:ERAT:CAT? 'ERAT_1,ERAT_2,ERAT_3'

:PLUG:ERAT:NEW
Syntax :PLUG:ERAT:NEW 'identifier'
Input 'identifier': Specify the measurement name.
Description This command creates a new error ratio measurement name identifier.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.
Example The following example creates an error ratio measurement name identifier called 'MyMeasurement':
:PLUG:ERAT:NEW 'MyMeasurement'

NOTE Creating multiple plugins using this command may slow down the GUI operations which may also result delay in remote programming. To prevent the plugin from opening automatically in the GUI, it is recommended to use "0" as a parameter input in this command.
Example- :PLUG:ERAT:NEW 'MyMeasurement',0

:PLUG:ERAT:DELeTe
Syntax :PLUG:ERAT:DELeTe 'identifier'
Input 'identifier': Specify the measurement name to delete.
Parameters
Description This command deletes a previously created error ratio measurement addressed by the measurement name identifier.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example
The following example deletes an error ratio measurement addressed by the measurement name identifier called 'MyMeasurement':

:\PLUG\ERAT\DEL 'MyMeasurement'

:\PLUG\ERAT\RES

Syntax      :\PLUG\ERAT\RES 'identifier'
Input       'identifier': Specify the measurement name to reset.
Parameters
Description  This command resets an error ratio measurement addressed by the measurement name identifier.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example
The following example resets an error ratio measurement addressed with the measurement name identifier called 'MyMeasurement':

:\PLUG\ERAT\RES 'MyMeasurement'

:\PLUG\ERAT\STAR

Syntax      :\PLUG\ERAT\STAR 'identifier'
Input       'identifier': Specify the measurement name to start.
Parameters
Description  This command starts an error ratio measurement addressed by the measurement name identifier.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example
The following example starts an error ratio measurement addressed with the measurement name identifier called 'MyMeasurement':

:\PLUG\ERAT\STAR 'MyMeasurement'
`:PLUGin:ERATio:STOP`

**Syntax**

`:PLUGin:ERATio:STOP 'identifier'`

**Input Parameters**

- `'identifier'`: Specify the measurement name to stop.

**Description**

This command stops an error ratio measurement addressed by the measurement name identifier.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

**Example**

The following example stops an error ratio measurement with the measurement name identifier called 'MyMeasurement':

`:PLUG:ERAT:STOP 'MyMeasurement'`

`:PLUGin:ERATio:SHOW:SGLegends`?

**Syntax**

`:PLUGin:ERATio:SHOW:SGLegends 'Identifier', <0|1|OFF|ON>


**Input Parameters**

- `'Identifier'`: 'MyMeasurement'

- `<0|1|OFF|ON>`: Specify the show/hide state for graph legends.

**Return Range**

0|1

**Description**

This command shows/hides the graph legends.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

**Example**

`:PLUG:ERAT:SHOW:SGL 'MyMeasurement', 1`

`:PLUGin:ERATio:SHOW:COMment`?

**Syntax**

`:PLUGin:ERATio:SHOW:COMment 'Identifier', <"Comment">

`:PLUGin:ERATio:SHOW:COMment? 'Identifier'

**Input Parameters**

- `'Identifier'`: Specify the measurement name. For example; 'MyMeasurement'

- `<"Comment">`: Enter the desired comment.
Description
This command is used to enter a comment for each measurement. This comment will be displayed on the Measurement History tab.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A, M8045A and M8046A.

Example
:PLUG:ERAT:SHOW:COM 'MyMeasurement', "Eye Measurement Example"
:PLUG:ERAT:SHOW:COM? 'MyMeasurement'
Eye Measurement Example

:PLUGIN:ERATio:GRAPH?

Syntax
:PLUGIN:ERATio:GRAPH? 'identifier'
[,]<DesiredWidth>,[,]<DesiredHeight>,[,]<CURR | WHIT>,[,]<PNG>,[,]1 | 0[,]"TabName"

Input Parameters
'identifier': Specify measurement name.

Other optional parameters are:
[,]<DesiredWidth>: Specify the desired width of the image.
[,]<DesiredHeight>: Specify the desired height of the image.
[,]<CURR | WHIT>: Specify whether the user wants to capture in current theme or wants to capture in white background.
[,]<PNG>: Specify the format of the image. The default format is PNG.
[,]1 | 0: Specify whether to capture the graph with legends or not.
[,]"TabName": Specify the tab name on which the graph is supposed to be captured.

Description
This query returns data of the image captured from the graph view of the plugin in the specified format.

Example
:PLUG:ERAT:GRAP? 'MyMeasurement',1000,800,CURR,PNG,1,'Graph'

:PLUGIN:ERATio:RUN:HISTory[:STATE]?

Syntax
:PLUGIN:ERATio:RUN:HISTory[:STATE] 'identifier' <0|1|ON|OFF>

Input Parameters
'identifier': Specify the measurement name.
Return Range  0|1|ON|OFF
Description  This command enables/disables the storage of error ratio measurement results addressed by the measurement name identifier. This query returns the current setting.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example  The following example enables storage of error ratio measurement results with the measurement name identifier called 'MyMeasurement':
:PLUG:ERAT:RUN:HIST 'MyMeasurement', 1

:PLUGin:ERATio:RUN:HISTory:CLEar

Syntax  :PLUGin:ERATio:RUN:HISTory:CLEar 'identifier'
Input  'identifier': Specify the measurement name.
Parameters
Description  This command deletes the storage of error ratio measurement history addressed by the measurement name identifier.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example  The following example deletes error ratio measurement history with the measurement name identifier called 'MyMeasurement':
:PLUGin:ERATio:RUN:HISTory:CLEar 'MyMeasurement'

:PLUGin:ERATio:RUN:LOG?

Input  'identifier': Specify the measurement name.
Parameters
Description  This command returns logs for the addressed measurement.
This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example  The following example returns logs for the measurement name identifier called 'MyMeasurement':

The format of log returned is in the following way:

#XY Message Log

where X denotes the length of digits for Y, Y denotes the number of characters in the log then followed by the log message.

For example:

#10 means, there is 1 digit after 1 and there is 0 character in the log.

#2492 means, there are 3 digits after 2 and there are 492 characters in the log message.

:PLUGin:ERATio:RUN:MESSage?


Input 'identifier': Specify the measurement name.

Description This query returns a string describing the state of an error ratio measurement addressed by the measurement name identifier. Possible states include NotStarted, Running, Finished, Error, or Stopped.

This SCPI is applicable for M8041A, M8051A, M8061, M8062A and M8046A.

Example The following example returns the state of an error ratio measurement with the measurement name identifier called 'MyMeasurement':

:PLUG:ERAT:RUN:MESS? 'MyMeasurement'
Running
:PLUGin:ERATio:RUN:PROGress?


Input  'identifier': Specify measurement name.

Parameters

Return Range  0.0 to 1.0

Description  This query returns a number in the range of 0.0 to 1.0 to indicate the progress of an error ratio measurement addressed by the measurement name identifier. A 0.0 indicates that the measurement has not started and 1.0 indicates the measurement is finished.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example  The following example returns the progress of an error ratio measurement with the measurement name identifier called 'MyMeasurement':

:PLUG:ERAT:RUN:PROG? 'MyMeasurement'

0.51

:PLUGin:ERATio:RUN[:STATus]?

Syntax  :PLUGin:ERATio:RUN[:STATus]? 'identifier'

Input  'identifier': Specify measurement name.

Parameters

Return Range  0|1

Description  This command returns the running status of an error ratio measurement addressed by the measurement name identifier. A 0 indicates the measurement is not running and a 1 indicates the measurement is running.

This SCPI is applicable for M8041A, M8051A, M8061A, M8062A and M8046A.

Example  The following example returns the running status of an error ratio measurement with the measurement name identifier called 'MyMeasurement':

:PLUG:ERAT:RUN 'MyMeasurement', 1
The M8020A/M8030A/M8040A Analyzer captures the data received from the device under test. The captured data bits are displayed in the pattern capture panel in binary or 8b/10b symbol coding. The received data is compared with the expected data and the errored bits/symbols are highlighted. The captured data can be saved for post processing. The maximum capture memory is 2 Gb. However, it also depends on the holdoff length which represents the amount of symbols in which the trigger events will be ignored.

This subnode has the following SCPI structure:

```
:PLUGin:CCAPture Subnode

The M8020A/M8030A/M8040A Analyzer captures the data received from the device under test. The captured data bits are displayed in the pattern capture panel in binary or 8b/10b symbol coding. The received data is compared with the expected data and the errored bits/symbols are highlighted. The captured data can be saved for post processing. The maximum capture memory is 2 Gb. However, it also depends on the holdoff length which represents the amount of symbols in which the trigger events will be ignored.

This subnode has the following SCPI structure:

```
```
This subnode has the following commands and subnodes:

Table 72

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELete[?]</td>
<td>:PLUGin:CCAPture:DELete on page 354</td>
</tr>
<tr>
<td>NEW[?]</td>
<td>:PLUGin:CCAPture:NEW on page 354</td>
</tr>
<tr>
<td>RESet[?]</td>
<td>:PLUGin:CCAPture:RESet on page 355</td>
</tr>
<tr>
<td>STARt[?]</td>
<td>:PLUGin:CCAPture:STARt on page 355</td>
</tr>
<tr>
<td>STOP[?]</td>
<td>:PLUGin:CCAPture:STOP on page 355</td>
</tr>
<tr>
<td>SAVE[?]</td>
<td>:PLUGin:CCAPture:SAVE on page 356</td>
</tr>
</tbody>
</table>

Subnodes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>FETCh</td>
<td>:PLUGin:CCAPture:FETCh Subnode on page 358</td>
</tr>
<tr>
<td>ACQuisition</td>
<td>:PLUGin:CCAPture:ACQuisition Subnode on page 364</td>
</tr>
<tr>
<td>RUN</td>
<td>:PLUGin:CCAPture:RUN Subnode on page 368</td>
</tr>
</tbody>
</table>
5 SCPI Command Reference

:PLUGin:CCAPture:CATalog?
Syntax :PLUGin:CCAPture:CATalog?
Description This command returns a list of all created compare and capture measurement names currently available for measuring.
Example Assume the following is a list of created compare and capture measurement names:
:PLUG:CCAP:NEW 'CCAP_1'
:PLUG:CCAP:NEW 'CCAP_2'
:PLUG:CCAP:NEW 'CCAP_3'
The command and returned list would look like the following:
:PLUG:CCAP:CAT?
"CCAP_1,CCAP_2,CCAP_3"

:PLUGin:CCAPture:DELete
Syntax :PLUGin:CCAPture:DELete 'identifier'
Input 'identifier': Specify the measurement name to delete.
Parameters Description This command deletes a previously created compare and capture measurement addressed by the measurement name identifier.
Example The following example deletes a compare and capture measurement addressed by the measurement name identifier called 'MyMeasurement':
:PLUG:CCAP:DEL 'MyMeasurement'

:PLUGin:CCAPture:NEW
Syntax :PLUGin:CCAPture:NEW 'identifier'
Input 'identifier': Specify the measurement name.
Parameters Description This command creates a new compare and capture measurement name identifier.
Example: The following example creates a compare and capture measurement name identifier called 'MyMeasurement':
:PLUG:CCAP:NEW 'MyMeasurement'

:PLUGin:CCAPture:RESet

Syntax: :PLUGin:CCAPture:RESet 'identifier'

Input: 'identifier': Specify the measurement name to reset.

Parameters

Description: This command resets a compare and capture measurement addressed by the measurement name identifier.

Example: The following example resets a compare and capture measurement addressed with the measurement name identifier called 'MyMeasurement':
:PLUG:CCAP:RES 'MyMeasurement'

:PLUGin:CCAPture:STARt

Syntax: :PLUGin:CCAPture:STARt 'identifier'

Input: 'identifier': Specify the measurement name to start.

Parameters

Description: This command starts a compare and capture measurement addressed by the measurement name identifier.

Example: The following example starts a compare and capture measurement addressed with the measurement name identifier called 'MyMeasurement':
:PLUG:CCAP:STAR 'MyMeasurement'

:PLUGin:CCAPture:STOP

Syntax: :PLUGin:CCAPture:STOP 'identifier'

Input: 'identifier': Specify the measurement name to stop.

Parameters

Description: This command stops the capturing of data and data is not uploaded into the measurement 'plug-in'.
**Example**

The following example stops a compare and capture measurement with the measurement name identifier called 'MyMeasurement':

```
:PLUG:CCAP:STOP 'MyMeasurement'
```

**:PLUGin:CCAPture:SAVE**

**Syntax**

```
:PLUGin:CCAPtue:SAVE 'identifier','Location','pattern-path'
[,overwrite-flag]
```

**Input Parameters**

- 'identifier': Specify the measurement name to save.
- 'Location': 'M*.DataIn1' or 'M*.DataIn2'
- '<pattern-path>': Patterns can be saved as data and are stored in three different locations:
  1. Local to current setting ("current:")
  2. Shared between settings ("shared:")
  3. Factory supplied standard patterns ("factory:"). These patterns are read only and cannot be modified.
- [,<overwrite-flag>]: ON | OFF | 0 | 1

Optional flag is a boolean parameter which specifies if an existing pattern file should be overwritten or not.

**Description**

The currently uploaded pattern is stored into the specified file on the hard disk.

**Example**

The following example saves a compare and capture measurement with the measurement name identifier called 'MyMeasurement':

```
```

**:PLUGin:CCAPtue:TRIGger:IMMediate[:ONCE]**

**Syntax**

```
:PLUGin:CCAPtue:TRIGger:IMMediate[:ONCE]
```

**Input Parameters**

- 'identifier': Specify the measurement name.
- 'Location': 'M*.DataIn1' or 'M*.DataIn2'

**Description**

This command simulates a stop event. Compare and capture is stopped and the recorded data is uploaded into the measurement.
Example: `PLUGin:CCAPture:TRIGger:IMMediate:ONCE 'MyMeasurement'`

`:PLUGin:CCAPture:SHOW:BIT[:PATTern][:VIEW][?]`


- **Input Parameters**
  - 'identifier': Specify the measurement name.
  - 'Mode': BIN|HEX

- **Return Range** BIN|HEX

- **Description**
  This command selects the bit pattern view mode viz. BIN|HEX.

- **Example**

`:PLUGin:CCAPture:SHOW:SYMBol[:PATTern][:VIEW][?]`


- **Input Parameters**
  - 'identifier': Specify the measurement name.
  - 'Mode': BIN|HEX|SYMBol

- **Return Range** BIN|HEX|SYMBol

- **Description**
  This command selects 8/10 bit pattern view mode viz. BIN|HEX|SYMBol.

- **Example**
This subnode has the following SCPI structure:

```
:FETCH
  :ERRor
    :COUNT?
  :PATTERN
    [:DATA]
    :HOLDoff?
    :DEPTH?
  :ERRor
    [:SYMBOL]
    :FIRST?
    :LAST?
  :RUN
  [:STATUS]?
  :ERRor:SYMBOL
  [:COUNT]?
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
</table>
### :PLUGin:CCAPture:FETCh:ERRor:COUNt?

**Syntax**

`:PLUGin:CCAPture:FETCh:ERRor:COUNt? 'identifier' [, 'Location']`

**Input Parameters**

'identifier': Specify the measurement name.

[,'Location']': 'M*.DataIn1' or 'M*.DataIn2'

**Description**

This query gives the error count of the captured data. If simply run without specifying the location, then it returns the results for the current run.

**Example**

`:PLUGin:CCAPture:FETCh:ERRor:COUNt? 'MyMeasurement', 'M1.DataIn1'

### :PLUGin:CCAPture:FETCh:PATTern:[DATA]:HOLDoff?

**Syntax**

`:PLUGin:CCAPture:FETCh:PATTern:[DATA]:HOLDoff? 'identifier' ['Location']`

**Input Parameters**

'identifier': Specify the measurement name.

[,'Location']': 'M*.DataIn1' or 'M*.DataIn2'

**Description**

This query returns the actual holdoff i.e. number of data bits captured before the trigger event. Returned value of this query is only valid when capture measurement is finished.

**Example**


Return: ('M1.DataIn1", ...)`
:PLUGin:CCAPture:FETCh:PATTern[:DATA]:DEPTh?

Syntax  :PLUGin:CCAPture:FETCh:PATTern[:DATA]:DEPTh? 'identifier' [,'Location']

Input  'identifier': Specify the measurement name

Parameters  [,'Location']: 'M*.DataIn1' or 'M*.DataIn2'

Description  This query returns the total number of data bits captured. Returned value of this query is only valid when capture measurement is finished.


Return: ('M1.DataIn1", ...)  

:PLUGin:CCAPture:FETCh:PATTern:ERRor[:SYMBol]:FIRST?

Syntax  :PLUGin:CCAPture:FETCh:PATTern:ERRor[:SYMBol]:FIRSt? 'identifier' [,'Location']

Input  'identifier': Specify the measurement name.

Parameters  [,'Location']: 'M*.DataIn1' or 'M*.DataIn2'

Description  This query fetches the first error symbol.

Example  :PLUGin:CCAPture:FETCh:PATTern:ERRor:FIRSt? 'MyMeasurement', 'M1.DataIn1'

Return: ('M1.DataIn1", ...)  

:PLUGin:CCAPture:FETCh:PATTern:ERRor[:SYMBol]:LAST?

Syntax  :PLUGin:CCAPture:FETCh:PATTern:ERRor[:SYMBol]:LAST? 'identifier' [,'Location']

Input  'identifier': Specify the measurement name.

Parameters  [,'Location']: 'M*.DataIn1' or 'M*.DataIn2'

Description  This query fetches the last error symbol.


Return: ('M1.DataIn1", ...)
:PLUGin:CCAPture:FETCh:RUN[:STATus]?

Syntax: :PLUGin:CCAPture:FETCh:RUN[:STATus]? 'identifier' ['Location']

Input Parameters:
- 'identifier': Specify the measurement name.
- ['Location']: 'M*.DataIn1' or 'M*.DataIn2'

Description: This query returns the capture measurement status for each analyzer location. Returned value can be any of the following:

- **STOP**: Capture measurement is stopped or not currently running. No valid data is available in capture memory.
- **STAR**: Capture measurement is starting. No valid data is available in capture memory.
- **ARM**: Capture measurement is started (Armed) and waiting for trigger. No valid data is available in capture memory.
- **TRIG**: Capture logic is triggered and data is being captured. Valid data will be available soon in capture memory.
- **UPL**: Data capturing is done and data is being uploaded and analyzed. Valid data is available in captured memory.
- **FIN**: Capture measurement is finished.

Return: ('M1.DataIn1', ...)
:PLUGin:CCAPture:FETCh:ERRor:SYMBol[:COUNT]?

**Syntax**
:PLUGin:CCAPture:FETCh:ERRor:SYMBol[:COUNT]? 'identifier'

**Input Parameters**
- **identifier**: Specify the measurement name.

**Description**
This query returns the symbol error counts per threshold level. The results are combined to an expression. An expression is comparable to a structure in a programming language. It's a set of parameters of different types separated by a comma and enclosed by parenthesis.

The returned expression contains the location, a timestamp and a set of symbol/bit counter values per threshold level. The returned expression contains counters according to the line coding. If the line coding is PAM4 then symbol counters are returned for four levels and in case of NRZ the bit counters are returned for two levels.

**Returned expression in case of PAM4**:

`("Location", TimeStamp, SymbolErrorRatio, ComparedSymbols, ErroneousSymbols, NLevels,(`

`(Level0, Level0SymbolErrorRatio, Level0ComparedSymbols, Level0ErroneousSymbols, (RecievedLevel0, RecievedLevel1, RecievedLevel2, RecievedLevel3)),`

`(Level1, Level1SymbolErrorRatio, Level1ComparedSymbols, Level1ErroneousSymbols, (RecievedLevel0, RecievedLevel1, RecievedLevel2, RecievedLevel3)),`

`(Level2, Level2SymbolErrorRatio, Level2ComparedSymbols, Level2ErroneousSymbols, (RecievedLevel0, RecievedLevel1, RecievedLevel2, RecievedLevel3)),`

`(Level3, Level3SymbolErrorRatio, Level3ComparedSymbols, Level3ErroneousSymbols, (RecievedLevel0, RecievedLevel1, RecievedLevel2, RecievedLevel3))

`))`

**Location**: It's a string referred to the analyzer location for which symbol errors are calculated.

**Timestamp**: As timestamp the UNIX epoch time is used; it is defined as elapsed seconds, milliseconds etc. since 01.01.1970 UTC.

**SymbolErrorRatio**: Symbol Error Ratio calculated from total errored and compared symbols.
**ComparedSymbols**: No of total compared symbols.

**ErroneousSymbols**: No of total erroneous symbols.

**NLevels**: No of levels compared i.e. 4 in case of PAM4 and 2 in case of NRZ.

**Level0-N**: Level no that specifies counters start for specific level 0 to 3 in case for PAM4 and 0 or 1 in case of NRZ.

**Level0-NSymbolErrorRatio**: Symbol error ratio for level N

**Level0-NComparedSymbols**: No of expected/compared symbols for level N

**Level0-NErroneousSymbols**: No of erroneous symbols for level N

**RecievedLevel0**: No of received Level 0 for expected Level N

**RecievedLevel1**: No of received Level 1 for expected Level N

**RecievedLevel2**: No of received Level 2 for expected Level N

**RecievedLevel3**: No of received Level 3 for expected Level N

**Example**: `:PLUGin:CCAPture:FETCh:ERRor:SYMBol:COUNt? 'Pattern Capture 1'

Return:

Line coding PAM4

("M1.DataIn1",1558690992.76551,0.000125,40000,5,4,((0,0.00016949,11800,2,(11798,2,0,0)),(1,0.0002439,8200,2,(0,8198,2,0)),(2,0.00009091,11000,1,(0,0,10999,0)),(3,0.0,9000,0,(0,0,0,9000))))

Line coding NRZ

("M1.DataIn1",1558690992.76551,0.0002,20000,4,2,(0.00002439,8200,2,(0,8198,2,0)),(2,0.00009091,11000,1,(0,0,10999,0)),(3,0.0,9000,0,(0,0,0,9000))))
:PLUGin:CAPture:ACQuisition Subnode

This subnode has the following SCPI structure:

:ACQUisition
  --:ALOCation[?]
  --:DEPTh[?]
  --:HOLDoff[?]
  --:SOURce[?]
  --:CIMemory[?]
  --:CAPOnly[?]

This subnode has the following commands:

Table 74

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DEPTh[?]</td>
<td>:PLUGin:CAPture:ACQuisition:DEPTh[?] on page 365</td>
</tr>
<tr>
<td>:HOLDoff[?]</td>
<td>:PLUGin:CAPture:ACQuisition:HOLDoff[?] on page 365</td>
</tr>
<tr>
<td>:SLOPe[?]</td>
<td>:PLUGin:CAPture:ACQuisition:SLOPe[?] on page 366</td>
</tr>
</tbody>
</table>
**:PLUGin:CCAPtue:ACQuisition:ALOCation[?]**

**Syntax**

```
:PLUGin:CCAPtue:ACQuisition:LOCation 'identifier','Location'

:PLUGin:CCAPtue:ACQuisition:LOCation? 'identifier'
```

**Input Parameters**

- `'identifier'`: Specify the measurement name.
- `['Location']`: 'M*.DataIn1' or 'M*.DataIn2'

**Description**

Specified the location the compare & capture measurement cooperates with. Examples for a location are 'M1.DataIn1', 'M3.DataIn2',... or a name of an existing group of DataIns.

This SCPI accepts error detector locations of M8041A, M8051A, M8062A, and error detector channels of all supported real-time oscilloscopes.

**Example**

```
:PLUGin:CCAPtue:ACQuisition:LOCation 'My Measurement', 'M1.DataIn1'
```

**:PLUGin:CCAPtue:ACQuisition:DEPTh[?]**

**Syntax**

```
:PLUGin:CCAPtue:ACQuisition:DEPTh 'identifier',<NRf>

:PLUGin:CCAPtue:ACQuisition:DEPTh? 'identifier'
```

**Input Parameters**

- `'identifier'`: Specify the measurement name.
- `Range 0 to 2147483648`

**Description**

This command configures how long the capture logic will write the capture memory after detecting the trigger event.

**Example**

```
:PLUGin:CCAPtue:ACQuisition:DEPTh 'My Measurement', 1
```

**:PLUGin:CCAPtue:ACQuisition:HOLDoff[?]**

**Syntax**

```
:PLUGin:CCAPtue:ACQuisition:HOLDoff 'identifier',<NRf>

:PLUGin:CCAPtue:ACQuisition:HOLDoff? 'identifier'
```

**Input Parameters**

- `'identifier'`: Specify the measurement name.
- `Range 0 to 2147483648`

**Description**

Specifies the amount of bits the trigger event is ignored after starting capturing of data.

**Example**

```
:PLUGin:CCAPtue:ACQuisition:HOLDoff 'My Measurement', 1
```
### :PLUGin:CAPture:ACQuisition:SOURce[?]

**Syntax**

```
:PLUGin:CAPture:ACQuisition:SOURce 'identifier', <CINA | CINB | ERRor | IMM>
```

```
:PLUGin:CAPture:ACQuisition:SOURce? 'identifier'
```

**Input Parameters**

- **'identifier'**: Specify the measurement name.

**Return Range**

CINA | CINB | ERRor | IMM

**Description**

Specifies the event for triggering the compare & capture measurement. CTRL IN A or CTRL IN B or an ERRor occurred can be chosen to trigger the capture recording. IMM can be used to capture data without any trigger.

**Example**

```
:PLUGin:CAPture:ACQuisition:SOURce 'My Measurement', CINA
```

### :PLUGin:CAPture:ACQuisition:SLOPe[?]

**Syntax**

```
:PLUGin:CAPture:ACQuisition:SLOPe 'identifier', <POS | NEG>
```

```
:PLUGin:CAPture:ACQuisition:SLOPe? 'identifier'
```

**Input Parameters**

- **'identifier'**: Specify the measurement name.

**Return Range**

POS | NEG

**Description**

If CINA or CINB is selected for triggering the capture process this command specifies the slope the trigger react with.

**Example**

```
:PLUGin:CAPture:ACQuisition:SLOPe 'My Measurement', POS
```

### :PLUGin:CAPture:ACQuisition:CIMemory[?]

**Syntax**

```
:PLUGin:CAPture:ACQuisition:CIMemory 'identifier', <ON|OFF|1|0>
```

```
:PLUGin:CAPture:ACQuisition:CIMemory? 'identifier'
```

**Input Parameters**

- **'identifier'**: Specify the measurement name.

**Return Range**

1 | 0
Description: This command allows the user to capture the data in memory.

Example:

```
:PLUG:CCAP:ACQ:CIM 'My Measurement', ON
:PLUG:CCAP:ACQ:CIM? 'My Measurement'
1
```

**:PLUGin:CCAPture:ACQuisition:CAPOnly[?]**

Syntax:

```
:PLUGin:CCAPture:ACQuisition:CAPOnly 'identifier', <ON|OFF|1|0>
:PLUGin:CCAPture:ACQuisition:CAPOnly? 'identifier'
```

Input Parameters:

- `'identifier'`: Specify the measurement name.
- `<ON | OFF | 1 | 0>`

Return Range: 1 | 0

Description: This command allows the user to capture data without any comparison. So use this command if you don't care about the comparison and also don't want to save the error bits.

Example:

```
:PLUG:CCAP:ACQ:CAPO 'My Measurement', ON
:PLUG:CCAP:ACQ:CAPO? 'My Measurement'
1
```
:PLUGin:CCAPture:RUN Subnode

This subnode has the following SCPI structure:

```
:RUN
  --:MESSage?
  --:PROGress?
  --:STATus?
  --:LOG?
```

This subnode has the following commands:

Table 75

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
</table>
:PLUGin:CCAPture:RUN:MESSage?

Syntax  :PLUGin:CCAPture:RUN:MESSage? 'identifier'

Input   'identifier': Specify the measurement name.

Parameters

Description This command returns a string describing the state of the compare & capture measurement. Possible states include NotStarted, Running, Stopped and Triggered.

Example The following example returns the state of a compare and capture measurement with the measurement name identifier called 'MyMeasurement':
:PLUG:CCAP:RUN:MESS? 'MyMeasurement'
Running

:PLUGin:CCAPture:RUN:PROGress?


Input   'identifier': Specify the measurement name.

Parameters

Return Range 0.0 to 1.0

Description This command returns a number in the range of 0.0 to 1.0 to indicate the progress of a compare and capture measurement addressed by the measurement name identifier. A 0.0 indicates that the measurement has not started and 1.0 indicates the measurement is finished.

Example The following example returns the progress of a compare and capture measurement with the measurement name identifier called 'MyMeasurement':
:PLUG:CCAP:RUN:PROG? 'MyMeasurement'
0.51
:PLUGin:CCAPture:RUN:STATus?

Syntax :PLUGin:CCAPture:RUN:STATus? 'identifier'

Input 'identifier': Specify the measurement name.

Parameters

Return Range 0|1

Description This command returns the running status of the compare & capture measurement. A 0 indicates the measurement is not running and a 1 indicates the measurement is running.

Example The following example returns the running status of a compare & capture measurement with the measurement name identifier called 'MyMeasurement':

:PLUG:CCAP:RUN? 'MyMeasurement'

:PLUGin:CCAPture:RUN:LOG?

Syntax :PLUGin:CCAPture:RUN:LOG? 'identifier'

Input 'identifier': Specify the measurement name.

Parameters

Description This command return the logs for the addressed measurement.

Example The following example return the logs for the measurement name identifier called 'MyMeasurement':


The format of log returned is in the following way:

#XY Message Log

where X denotes the length of digits for Y, Y denotes the number of characters in the log then followed by the log message.

For example:

#10 means, there is 1 digit after 1 and there is 0 character in the log.

#2492 means, there are 3 digits after 2 and there are 492 characters in the log message.
Advanced Measurement Package

The Advanced Measurement Package provides the following measurements when it is installed on M8070B System Software:

- Output timing measurement
- Output level measurement
- Jitter tolerance measurement
- Eye diagram measurement
- Parameter sweep

**NOTE**

Advanced Measurement Package requires a valid license for activation.

---

For more details on the Advanced Measurement Package and its remote programming, please refer to *M8000 Series of Advanced Measurement Package User Guide* and Online Help.

Error Distribution Analysis Package

The Error Distribution Analysis package provides the following measurement when it is installed on M8070B System Software:

- Error Distribution Analysis

**NOTE**

Error Distribution Analysis Package requires a valid license for activation.

---

For more details on the Error Distribution Analysis Package and its remote programming, please refer to *M8000 Series of Error Distribution Analysis Package User Guide* and Online Help.
SYSTem Subsystem

The SYSTem subsystem is used for general system functions.
This subsystem has the following SCPI structure:
This subsystem has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:INSTrument:RESet</td>
<td>:SYSTem:INSTrument:RESet on page 375</td>
</tr>
<tr>
<td>:INSTrument:GROUP[:LIST]?</td>
<td>:SYSTem:INSTrument:GROUP[:LIST]? on page 376</td>
</tr>
<tr>
<td>:INSTrument:LTRaining[:LOG]?</td>
<td>:SYSTem:INSTrument:LTRaining[:LOG]? on page 377</td>
</tr>
<tr>
<td>:INSTrument:LTRaining[:LOG]:CLEar</td>
<td>:SYSTem:INSTrument:LTRaining[:LOG]:CLEar on page 378</td>
</tr>
<tr>
<td>:INFormation[:INSTrument]:IDENtifier?</td>
<td>:SYSTem:INFormation[:INSTrument]:IDENtifier? on page 380</td>
</tr>
<tr>
<td>:DCINterface[:PROGram]:LOAD</td>
<td>:SYSTem:DCINterface[:PROGram]:LOAD on page 384</td>
</tr>
<tr>
<td>:DCINterface[:PROGram]:UNLoad</td>
<td>:SYSTem:DCINterface[:PROGram]:UNLoad on page 385</td>
</tr>
<tr>
<td>Name</td>
<td>Description under</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>:DCINterface:DEVice:CONNnect</td>
<td>:SYSTem:DCINterface:DEVice:CONNnect(?) on page 386</td>
</tr>
<tr>
<td>:RESet:INSTrument</td>
<td>:SYSTem:RESet:INSTrument on page 389</td>
</tr>
<tr>
<td>:SYSTem:MODule:UPDate:AUTomatic</td>
<td>:SYSTem:MODule:UPDate:AUTomatic on page 390</td>
</tr>
<tr>
<td>:SYSTem:MODule:UPDate:PENDing</td>
<td>:SYSTem:MODule:UPDate:PENDing on page 390</td>
</tr>
</tbody>
</table>
**:SYSTem:INStrument:RESet**

**Syntax**

**:SYSTem:INStrument:RESet**

**Description**

This command resets the instrument state to its default values. This command does not influence the SCPI status system as *RST does.

**Example**

:SYST:INST:RES

---

**:SYSTem:INStrument:GROup:DEFine**

**Syntax**

**:SYSTem:INStrument:GROup:DEFine**

'GroupName','identifier','identifier','identifier','...'  

**Input Parameters**

GroupName: Assign a name to this group of location identifiers.  
identifier: Define the group of location identifiers.

**Description**

This command defines a group of location identifiers under one group name. All properties belonging to these location identifiers can be addressed simultaneously using the group name.

**Example**

VOLT:AMPL 'Outputs',0.05

---

**NOTE**

If you define a group name that already exists, the existing name with its group of defined location identifiers will be redefined without notification.

---

**:SYSTem:INStrument:GROup:DELeTe**

**Syntax**

**:SYSTem:INStrument:GROup:DELeTe** 'GroupName'

**Input Parameters**

GroupName: Specify group name to delete.

**Description**

This command deletes the specified group.

**Example**

:SYST:INST:GRO:DEL 'Outputs'
:SYSTem:INStrument:GROup:MEMBer?
Syntax  :SYSTem:INStrument:GROup:MEMBer? 'GroupName'
Input   'GroupName': Specify group name.
Parameters
Description This command returns the comma separated list of member names contained within the specified group.
Example  :SYST:INST:GRO:MEMB? 'Outputs'
          "M1.DataOut1","M1.DataOut2"

:SYSTem:INStrument:GROup[:LIST]?
Syntax  :SYSTem:INStrument:GROup[:LIST]?
Description This command returns a comma separated list of all group names.
Example  :SYST:INST:GRO?
          "Outputs"
:SYSTem:INSTrument:LTRaining[:LOG]?

Syntax :SYSTem:INSTrument:LTRaining[:LOG]?

Description This command returns accumulated Link Training logs for the selected DataOut locations in text format.

Example :SYST:INST:LTR:LOG?


<table>
<thead>
<tr>
<th>State</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect.Active</td>
<td>1.04864 ms</td>
</tr>
<tr>
<td>Polling.Active</td>
<td>2.064992 ms</td>
</tr>
<tr>
<td>Polling,Configuration</td>
<td>152.112 us</td>
</tr>
<tr>
<td>Configuration.Linkwidth.Start</td>
<td>2.896 us</td>
</tr>
<tr>
<td>Configuration.Linkwidth.Accept</td>
<td>304 ns</td>
</tr>
<tr>
<td>Configuration.Lanenum.Wait</td>
<td>2.8 us</td>
</tr>
<tr>
<td>Configuration.Lanenum.Accept</td>
<td>544 ns</td>
</tr>
<tr>
<td>Configuration.Complete</td>
<td>3.696 us</td>
</tr>
<tr>
<td>Configuration.Idle</td>
<td>256 ns</td>
</tr>
<tr>
<td>L0</td>
<td>224 ns</td>
</tr>
<tr>
<td>Recovery.RcvrLock</td>
<td>3.52 us</td>
</tr>
<tr>
<td>Recovery.RcvrCfg</td>
<td>2.464 us</td>
</tr>
<tr>
<td>Recovery.Speed</td>
<td>8.464 us</td>
</tr>
<tr>
<td>Recovery.RcvrLock</td>
<td>448 ns</td>
</tr>
<tr>
<td>Recovery.Equalization.Phase1</td>
<td>2.20608 ms</td>
</tr>
<tr>
<td>Recovery.Equalization.Phase2</td>
<td>21.033456 ms</td>
</tr>
<tr>
<td>Recovery.Equalization.Phase3</td>
<td>1.904 us</td>
</tr>
<tr>
<td>Recovery.RcvrLock</td>
<td>2.016 us</td>
</tr>
<tr>
<td>Recovery.RcvrCfg</td>
<td>656 ns</td>
</tr>
<tr>
<td>Recovery.Idle</td>
<td>112 ns</td>
</tr>
<tr>
<td>Loopback.Entry</td>
<td>2.816 us</td>
</tr>
<tr>
<td>Loopback.Active</td>
<td>-</td>
</tr>
</tbody>
</table>

BERT Tx Equalization

<table>
<thead>
<tr>
<th>Accept</th>
<th>PresetNumber</th>
<th>PreCursor</th>
<th>MainCursor</th>
<th>PostCursor</th>
<th>FullSwing</th>
<th>LowFrequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>P7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>True</td>
<td>P4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>8</td>
</tr>
</tbody>
</table>

DUT Tx Equalization

<table>
<thead>
<tr>
<th>Event</th>
<th>Accept</th>
<th>PresetNumber</th>
<th>PreCursor</th>
<th>MainCursor</th>
<th>PostCursor</th>
<th>FullSwing</th>
<th>LowFrequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>-</td>
<td>P7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>40</td>
<td>12</td>
</tr>
<tr>
<td>Target</td>
<td>-</td>
<td>P4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>40</td>
<td>12</td>
</tr>
<tr>
<td>Reported True</td>
<td>P4</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>40</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
:SYSTem:INStrument:LTRaining[:LOG]:CLEar

Syntax  :SYSTem:INStrument:LTRaining[:LOG]:CLEar
Description  This command clears the Link Training logs.
Example  :SYST:INST:LTR:LOG:CLE

:SYSTem:INStrument:LTRaining[:LOG]:BLOOk?

Syntax  :SYSTem:INStrument:LTRaining[:LOG]:BLOck?
Description  The content of the log file is returned as definite block. (see :SYSTem:INStrument:LTRaining[:LOG]?) Control character (e.g. Line Feed) are contained in the log file. This can interrupt the transfer to the host computer. A definite block is used to transfer data streams that can contain special characters. see SCPI language definition or IEEE 488.2 -BLOCK PROGRAM DATA.
Example  :SYSTem:INStrument:LTRaining:LOG:BLOck?
:SYSTem:INStrument:LTRaining:EQUalization[:STATus]?

**Syntax**

:SYSTem:INStrument:LTRaining:EQUalization[:STATus]? 'identifier'

**Input Parameters**

'-identifier': 'M*.DataOut1' or 'M*.DataOut2'

**Description**

This query returns the Dut transmitter equalization information from the link up block. The results are combined to an expression. An expression is comparable to a struct in a programming language. It's a set of parameter of different types separated by a comma and enclosed by parenthesis.

The returned expression contains the location and a set of values.

("Location", Accepted, PreCursor, MainCursor, PostCursor, FullSwing, LowFrequency)

Location: It's a string referred to the place the Dut tx equalization information is located. In this case its either M*.DataOut1 or M*.DataOut2. This parameter is important if a group name is used as 'identifier'.

Accepted: Requested coefficients are accepted or rejected, either 0 or 1.

PreCursor: Received pre-cursor coefficient value during Phase 2 or 3 of the equalization. If no data is present, 9.91 E 37 is returned.

MainCursor: Received data-cursor coefficient value during Phase 2 or 3 of the equalization. If no data is present, 9.91 E 37 is returned.

PostCursor: Received post-cursor coefficient value during Phase 2 or 3 of the equalization. If no data is present, 9.91 E 37 is returned.

FullSwing: Full swing value received during Phase 1 of the equalization. If no data is present, 9.91 E 37 is returned.

LowFrequency: Low Frequency value received during Phase 1 of the equalization. If no data is present, 9.91 E 37 is returned.

If this query is sent to a 'group' of inputs the response will be expanded by additional sets of counted values. The expression contains for every input belonging to a group a separate set of values starting with the location.

**Example**

:SYSTem:INFormation[:INSTrument]:IDENtifier?

Syntax: :SYSTem:INFormation[:INSTrument]:IDENtifier?

Description: This command returns a list of all identifiers defined in the system.

Example: :SYST:INF:IDEN?


:SYSTem:INFormation[:INSTrument]:LOCations?

Syntax: :SYSTem:INFormation[:INSTrument]:LOCations?

Description: This command returns a list of all locations defined in the system.

Example: :SYST:INF:LOC?

(ClkGen,System,RefClkOut,TrigOut,SysOutA,SysOutB,CtrlOutA,CtrlOutB,DataIn1,DataIn2,DataOut1,DataOut2,Simulation1,Simulation2,MuxMode,ElectIdleIn)

:SYSTem:INFormation[:INSTrument]:GROups:USER?

Syntax: :SYSTem:INFormation[:INSTrument]:GROups:USER?

Description: This command returns a list of all user defined parameter group names currently in use by the system.

Example: :SYST:INF:GRO:USER?

(Outputs,Inputs)

:SYSTem:INFormation[:INSTrument]:DETails subnode

The following commands provide information about a specific part of the instrument.

The returned information contains
- Identifier
- Product number
- Connection status (Online/Offline)
• List of all options
• User friendly name of the instrument
• Serial number
• VISA resource string
• Software Version
• All the above information for sub-components like remote heads, or external clock recovery modules

:SYSTem:INFormation[:INSTrument]:DETails:JSON?

Syntax :SYSTem:INFormation[:INSTrument]:DETails:JSON? 'Identifier'

Description This command returns the instrument details in .JSON (JavaScript Object Notation) data structure.

Input Parameters Identifier - Any module name. For example; 'M1', 'M2'

Return Value Returns the sequence of following values:
Identifier, ProductNumber, Connection, Options, FriendlyName of module, SerialNumber, VisaResource, SoftwareVersion, Subcomponents

Example :SYST:INF:INST:DET:JSON? 'M2'
{"Identifier":"M2","ProductNumber":"M8046A","Connection":"Offline","Options:['A32','A64','OP3','OP6','O9A','U64','UP3','UP6','UA9'],"FriendlyName":"M8046A 64GBd Error Analyzer","SerialNumber":"DE5250000003","VisaResource":"OFFLINE::211::INSTR","SoftwareVersion":"5.0.125.1","Subcomponents":[]}
Return Value

Returns the instrument details value in the following sequence:
(identifier, product-number, connection-status, (installed-options),
friendly-name, serial-number, VISA-resource, software-version,
((sub-component-info-1), (sub-component-info-2), ...
(sub-component-info-n))

Example:
:M:SYSTem:INFormation[:INStrument]:DETaiIs:VAL? 'M2'
('M2','M8046A','Offline','A32,A64,0P3,0P6,0A9,U64,UP3,UP6,UA9','M8046
A 64GBd Error Analyzer','DE5250000003','OFFLINE::211::INSTR','5.0.125.1',())

:SYSTem:INFormation[:INStrument]:DETaiIs:XML?

Syntax: :SYSTem:INFormation[:INStrument]:DETaiIs:XML? 'Identifier'

Description
This command returns the instrument details in XML format.

Input Parameters
Identifier - Any module name. For example; 'M1', 'M2'

Return Value
Returns the following instrument details in XML format:
Identifier, Product Number, Connection type, License options, friendly
name of module, Serial Number, Visa resource, Software version,
Subcomponents

Example:
:M:SYSTem:INFormation[:INStrument]:DETaiIs:XML? 'M2'
<?xml version="1.0" encoding="utf-16"?>
<DetailedComponentInfo xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <Identifier>M2</Identifier> <ProductNumber>M8046A</ProductNumber> <Connection>Offline</Connection> <Options>A32,A64,0P3,0P6,0A9,U64,UP3,UP6,UA9</Options> <FriendlyName>M8046A 64GBd Error Analyzer</FriendlyName> <SerialNumber>DE5250000003</SerialNumber> <VisaResource>OFFLINE::211::INSTR</VisaResource> <SoftwareVersion>5.0.125.1</SoftwareVersion> <Subcomponents/>
</DetailedComponentInfo>
**SYSTem:INFormation:LiCenses:HOST[:LIST]?**

**Syntax**
SYSTem:INFormation:LiCenses:HOST[:LIST]?

**Description**
This command lists the description of all the host licenses supported by the device.

**Example**
SYSTem:INFormation:LiCenses:HOST?

((PCSERNO,FH34484899,("M8070A-CAL","CAL","1.0","System Software for M8000 Series, internal-use only, transportable perpetual license","Installed","Perpetual","12/31/9999 11:59:59 PM",""))
(M8070ADEM," DEM","1.0","Demo license for M8000 Series, demo-use only, transportable perpetual license","Installed","Perpetual","12/31/9999 11:59:59 PM",""))
(M8070ADVB,"","","Advanced Measurement Package for M8000 Series BERT Test Solutions","Not installed","","12/31/9999 11:59:59 PM",""))
(M8070EDAB,"","","Error Distribution Analysis Package for M8000 Series BERT Test Solutions","Not installed","","12/31/9999 11:59:59 PM",""))}
:SYSTem:INFormation:LICenses:MODules[:LIST]?

Syntax :SYSTem:INFormation:LICenses:MODules[:LIST]?

Description This command lists the description of all the module licenses supported by the device.

Example :SYSTem:INFormation:LICenses:MODules:LIST?

:SYSTem:COMMunicate:TCPip:CONTrol?

Syntax :SYSTem:COMMunicate:TCPip:CONTrol?

Description This command implements SCPI over TCP with Control Connection.

The Keysight Automation Advantage (AAA) standard requires that instruments that support SCPI also support SCPI traffic over TCP with a control connection. This control channel is set up by sending SYSTem:COMMunicate:TCPip:CONTrol? to the instrument. To use the control port:

- Open a socket connection to port 5025, the standard SCPI- over-sockets port.
- Over that socket connection, issue the SYST:COMM:TCP:CONT? query to retrieve the control port.
- Open a second socket connection to the control port.

Now you can send control commands: for example, send "DCL\n" on the control port to issue a device clear.

Example :SYST:COMM:TCP:CONT?

:SYSTem:DCINterface[:PROGram]:LOAD

Syntax :SYSTem:DCINterface[:PROGram]:LOAD "PathToPythonProgramFile"

Description This command loads the given Python script into the DUT control interface. This DUT control interface program can be used to read the bit error rate of an externally connected DUT and provide these to the M8070B software as if the counters are part of the instrument. These values can then be used by the built in Bit Error Ratio measurement or Jitter Tolerance measurement.

In addition the script can implement functions that extend the measurements in order to adapt the measurement to the specific test setup and device under test.
Python scripts can be part of settings and shared areas. The addressing scheme looks like:

"Factory/HookExample.py" // Read only factory shared area
"Shared/HookExample.py" // User shared area
"Current/HookExample.py" // Current setting

Example :SYSTem:DCINterface:PROGram:LOAD
"factory/DutCounterIntegration.py"

:SYSTem:DCINterface[:PROGram]:UNLoad

Syntax :SYSTem:DCINterface[:PROGram]:UNLoad

Description This command unloads the currently used DUT Control Interface script and all the resources that the script integrated into the system.

Example :SYSTem:DCINterface:PROGram:UNLoad

:SYSTem:DCINterface[:PROGram]:SELected?

Syntax :SYSTem:DCINterface[:PROGram]:SELected?

Description This query returns the DUT Control Interface script that is currently in use. If the DUT Control Interface is not using a script, an empty string is returned.

Example :SYSTem:DCINterface:PROGram:SELected?

:SYSTem:DCINterface:EXEcute[:INIT]

Syntax :SYSTem:DCINterface:EXEcute[:INIT] 'Location', ['ArgumentsForInit']

Input

Parameters

Description This command matches the DUT Control Interface hook function DUT_Init(location, argument).

Sending this command results in executing the function DUT_Init with the given parameters if it is implemented by the script that is currently in use (see syst:dcin:load).

The Location argument shall be one of the locations defined by the scripts DUT_getLocations, but as full system location identifier.
For example, if `DUT_getLocations` defines the locations "Lane1" and "Lane2", then valid Location arguments are "DCI.Lane1" and "DCI.Lane2"

The argument string is not checked anyway, but simply passed to the script implementation of `DUT_Init`.

Example

```
:SYSTem:DCINterface:execute:init "DCI.Lane1"
```

`:SYSTem:DCINterface:DEVice:CONNect[?]`

**Syntax**

`:SYSTem:DCINterface:DEVice:CONNect 'identifier', <ON|OFF|1|0>

`:SYSTem:DCINterface:DEVice:CONNect 'identifier'?

**Input Parameters**

- `'identifier'`: "DCI.Control"

**Description**

This command relates to the DUT Control Interface hook functions `DUT_connect` and `DUT_disconnect`.

When these functions are implemented by the currently used script (see `syst:dcin:load`), then this command is controlling the Connection parameter under the function `Device` for the Control location of the device under test.

The location argument must be the Control location as full system location identifier as displayed in the parameter editor (e.g. "DCI.Control").

This query returns the current setting.

Example

```
:SYSTem:DCINterface:DEVice:CONNect 'DCI.Control', 1
```
**:SYSTem:DCINterface:DEVice:MODE[?]**

**Syntax**

:SYSTem:DCINterface:DEVice:MODE 'identifier', 'mode'

:SYSTem:DCINterface:DEVice:MODE? 'identifier'

**Input Parameters**

- 'identifier': 'DCI.Control'
- 'mode': Individual modes or configurations.

**Description**

This command relates to the DUT Control Interface hook functions DUT_getDeviceModes and DUT_setDeviceMode.

When these functions are implemented by the currently used script (see syst:dcin:load), then this command is controlling the Mode parameter under the function Device for the Control location of the device under test.

The location argument must be the Control location as full system location identifier as displayed in the parameter editor (e.g. "DCI.Control").

The mode argument must be one of the strings returned by DUT_getDeviceModes.

This can be used to control configurations that are valid for the whole device under test.

When sending this command in one SCPI transaction with :syst:dcin:lane:mode, then the device mode hook (DUT_setDeviceMode) will be executed before the lane specific mode (DUT_setLaneMode).

This query returns the current setting.

**Example**

:SYSTem:DCINterface:DEVice:MODE 'DCI.Control','Configuration1'
:SYSTem:DCINterface:LANE:MODE(?)

Syntax
:SYSTem:DCINterface:LANE:MODE "Location", "mode"
:SYSTem:DCINterface:LANE:MODE? "Location"

Input
'Location': '*.Lane*'

Parameters
'mode': Individual modes or configurations.

Description
This command relates to the DUT Control Interface hook functions DUT_getLaneModes and DUT_setLaneMode.

When these functions are implemented by the currently used script (see syst:dcin:load), then this command is controlling the Mode parameter under the function Lane for the given lane of the device under test.

The location argument must be the Control location as full system location identifier as displayed in the parameter editor (e.g. "DCI.Lane1").

The mode argument must be one of the strings returned by DUT_getLaneModes.

This can be used to control configurations that are valid for an individual lane or channel of the device under test.

When sending this command in one SCPI transaction with :syst:dcin:dev:mode, then the device mode hook (DUT_setDeviceMode) will be executed before the lane specific mode (DUT_setLaneMode).

This query returns the current setting.

Example
:SYSTem:DCINterface:LANE:MODE 'DCI.Lane1','MODE1'
```plaintext
:SYSTem:RESet[:ALL]
Syntax  :SYSTem:RESet[:ALL] [HARD | SOFT]
Input   [HARD | SOFT]
Parameters
Description  This command resets the instrument.
              If SOFT is passed the instrument will load its default values and defined
              'Groups' in the GUI will be untouched.
              If HARD is passed the instrument will load the factory setting and all
              'Group' definitions will be cleared.
              Independent of passing HARD or SOFT all plug-ins (measurements) are
              closed.
Example     :SYSTem:RESet:ALL SOFT

:SYSTem:RESet:INSTrument
Syntax      :SYSTem:RESet:INSTrument [HARD | SOFT]
Input       [HARD | SOFT]
Parameters
Description  This command resets the instrument.
              If SOFT is passed the instrument will load its default values and defined
              'Groups' in the GUI will be untouched.
              If HARD is passed the instrument will load the factory setting and all
              'Group' definitions will be cleared.
Example      :SYSTem:RESet:INSTrument SOFT

:SYSTem:MODule:UPDate:APPLy
Syntax      :SYSTem:MODule:UPDate:APPLy "Location"
Input       <identifier> - "Location", e.g. "M1.System"
Parameters
Description  This command applies the currently configured software state into the
              module.
              This command is applicable for M8195A, M8196A & M8154A.
```
>:SYStem:MODule:UPDate[:AUTOmatic][:?]  
**Syntax**:  
>:SYStem:MODule:UPDate[:AUTOmatic] "Location", <ON|OFF|1|0>  
>:SYStem:MODule:UPDate[:AUTOmatic]? "Location"  
**Input Parameters**:  
<identifier> - "Location", e.g. "M1.System"  
<ON|OFF|1|0>  
**Description**:  
This command automatically applies the changes made to the instrument. When the Automatic Module Update option is disabled, the time intensive operation’s execution defers for parameter changes. Depending on the module type, this may disable some or all the property updates. Additionally, it can also disable the dynamic parameter limit calculations, which can trigger the “Auto Correct Confirmation” dialog when the module updates.  
This query returns the current settings.  
This command is applicable for M8195A, M8196A & M8154A.  
**Example**:  

>:SYStem:MODule:UPDate:PENDing[:?]  
**Syntax**:  
>:SYStem:MODule:UPDate:PENDing <identifier>, <ON|OFF|1|0>  
>:SYStem:MODule:UPDate:PENDing? <identifier>  
**Input Parameters**:  
<identifier> - "Location", e.g. "M1.System"  
**Description**:  
This applied the pending changes that are not yet sent to the hardware or otherwise not updated.  
This query returns the current settings.  
This command is applicable for M8195A, M8196A & M8154A.  
**Example**:  
DATA Subsystem

The DATA subsystem is used to select patterns, define symbol parameters, define sequence blocks and loops and synchronize the pattern.

This subsystem has the following SCPI structure:

```
:DATA
  :PATTern
  ...
  :SEQuence
  ...
  :SYNC
  ...
  :LINecoding
  ...
  :FEC
  ...
```

The subsystem has the following subnodes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:PATTern</td>
<td>:DATA:PATTern Subnode on page 392</td>
</tr>
<tr>
<td>:SEQuence</td>
<td>:DATA:SEQuence Subnode on page 400</td>
</tr>
<tr>
<td>:SYNC</td>
<td>:DATA:SYNC Subnode on page 409</td>
</tr>
<tr>
<td>:LINecoding</td>
<td>:DATA:LINecoding Subnode on page 411</td>
</tr>
<tr>
<td>:FEC</td>
<td>:DATA:FEC Subnode on page 415</td>
</tr>
</tbody>
</table>
:DATA:PATTern Subnode

For additional information about patterns, refer to Working with Patterns on page 36.

This subnode has the following SCPI structure:

```
:DATA
 - PATTern
   - :USE[?]
   - :IDATa[?]
   - :BLENgt?h
   - :CATalog?
   - :CINFormation?
   - :DESCription[?]
   - :DELe.te
   - :PACKed[?]
   - :IMPort
   - :EXP ort
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:USE[?]</td>
<td>:DATA:PATTern:USE[?] on page 393</td>
</tr>
<tr>
<td>:IDATa[?]</td>
<td>:DATA:PATTern:IDATa[?] on page 394</td>
</tr>
<tr>
<td>:DESCription[?]</td>
<td>:DATA:PATTern:DESCription[?] on page 396</td>
</tr>
<tr>
<td>:DELe.te</td>
<td>:DATA:PATTern:DELe.te on page 396</td>
</tr>
<tr>
<td>:IMPort</td>
<td>:DATA:PATTern:IMPort on page 397</td>
</tr>
<tr>
<td>:EXP ort</td>
<td>:DATA:PATTern:EXP ort on page 399</td>
</tr>
</tbody>
</table>
:DATA:PGM:USE[?]

Syntax

:DATA:PGM:USE <pattern-name>,<num-symbols>,<symbol-coding>
[,<mask-used>[,<squelch-used>,<error-used>]]

:DATA:PGM:USE? <pattern-name>

Input Parameters

<pattern-name>: Name of the pattern. Name consists of root node
("factory:“, “shared:“, or “current:“) plus folder names separated by ‘/’ plus
pattern name.

<num-symbols>: Length of the pattern in number of symbols.

<symbol-coding>: BIT|B8B10|B128B130|B128B132 for bit coding, 8b10b
coding, 128/130 coding or 128/132 coding. Default is BIT.

<mask-used>: 0|1|OFF|ON to specify if mask attribute is used. Default is
OFF. Not using mask is mostly useful for bit coding to save space.

<squelch-used>: 0|1|OFF|ON to specify if squelch attribute is used. Default
is OFF. Not using squelch is mostly useful for bit coding to save space.

<error-used>: 0|1|OFF|ON to specify if error attribute is used. Default is
OFF. Error Attribute is used for captured patterns. When <error-used> is
ON, <mask-used> and <squelch-used> must be OFF.

Return Range <num-symbols>, <symbol-coding>, <mask-used>,
<squelch-used>,<error-used>

Description

This command creates a new pattern. If a pattern exists with the same
name, it is overwritten.

Example

:DATA:PGM:USE "current:demo",20
:DATA:PGM:USE "current:demo8b10",5,b8b10
:DATA:PGM:USE "current:demo128b130",5,b128b130
:DATA:PGM:USE "current:demo128b132",5,b128b132
:DATA:PGM:USE? "current:demo"
5,BIT,0,0,0
:DATA:PGM:USE? "factory:SATA/LTDP_short_b8b10"
512,B8B10,1,1,0
:**DATA:**PATTern:**DATa[?]**

**Syntax**

```plaintext
:DATA:PATTern:DATa <pattern-name>,<bit-offset>,<num-bits>,<data-block>

:DATA:PATTern:DATa? <pattern-name>,<bit-offset>,<num-bits>
```

**Input Parameters**

- `<pattern-name>`: Name of the pattern. Name consists of root node ("shared:" or "current:" plus folder names separated by "/" plus pattern name. The "factory:" root node is not allowed here, as these patterns are read only and cannot be modified.

- `<bit-offset>`: Offset into the pattern bit sequence. Bit offset is used, to allow modifications within symbols, for example just modifying a single attribute bit.

- `<num-bits>`: Number of bits to modify. Specified in bits to allow modifications within symbols, e.g. just modifying a single attribute bit.

- `<data-block>`: Bit sequence as definite length block. Number of bits encoded in one byte is specified by the :DATA:PATT:PACK command.

**Return Range**

- `<pattern-name>`, `<bit-offset>`, `<num-bits>`

**Description**

This command is used to modify a pattern.

**Example**

The following example enables the scrambler in the first symbol of an 8b10b coded pattern.

```plaintext
:DATA:PATT:IDAT 'current:demo8b10b',12,1,#111
```

:**DATA:**PATTern:**BLENgth?**

**Syntax**

```plaintext
:DATA:PATTern:BLENgth? <pattern-name>
```

**Input Parameters**

- `<pattern-name>`: Name of the pattern. Name consists of root node ("factory:" or "shared:" or "current:" plus folder names separated by "/" plus pattern name.

**Return Range**

- `<num-bits>`

**Description**

This command returns the pattern length in bits of a specified pattern.

**Example**

```plaintext
```
:DATA:PATTern:CATalog?
Syntax :DATA:PATTern:CATalog? <pattern-path>
Input <pattern-path>: Location of patterns.
Parameters
Description This command returns a list of patterns found at the specified path.
Example :DATA:PATT:CAT? "factory:XAUI"
"CJPAT_bit,CRPAT_b8b10,CRPAT_bit"

:DATA:PATTern:CINFormation?
Syntax :DATA:PATTern:CINFormation?
<symbol-coding>[,<mask-used>[,<squelch-used>]]
Input <symbol-coding>: BIT|B8B10|B128B130|B128B132 for bit coding, 8b10b coding, 128/130 coding or 128/132 coding.
Parameters <mask-used>: 0|1|OFF|ON to specify if mask attribute is used. Default is OFF. Not using mask is mostly useful for bit coding to save space.
<squelch-used>: 0|1|OFF|ON to specify if squelch attribute is used. Default is OFF. Not using squelch is mostly useful for bit coding to save space.
Return Range <symbol-coding>, <mask-used>, <squelch-used>
Description This command returns coding information.
Example :DATA:PATT:CINF? B8b10,ON,ON
:DATA:PATTern:DESCription[?]

Syntax :DATA:PATTern:DESCription <pattern-name>,<description>
        :DATA:PATTern:DESCription? <pattern-name>

Input <pattern-name>: Name of the pattern. Name consists of root node
      Parameters ("factory:", "shared:“, or "current:“) plus folder names separated by ‘/’ plus
      pattern name; like "shared:SATA/LTDP_short_b8b10".
      <description>: Enter a pattern description.

Return Range <description>

Description This command is used to add a description to the specified pattern. This
SCPI doesn’t allow adding description to factory patterns.

Example Create a new pattern by the name "bit_pattern" and save it to "shared"
      memory location. Now, use the following example to set "sample bit
      pattern" as the description.
      :DATA:PATT:DESC "shared:bit_pattern","sample bit pattern"

:DATA:PATTern:DELeTe

Syntax :DATA:PATTern:DELeTe <pattern-name>

Input <pattern-name>: Name of the pattern. Name consists of root node
      Parameters ("factory:“, "shared:“, or "current:“) plus folder names separated by ‘/’ plus
      pattern name.

Description This command deletes the specified pattern. A factory pattern can’t be
      deleted.

Example :DATA:PATT:DEL "shared:SATA/LTDP_short_b8b10"
:DATA:PATTern:PACKed[
]
Syntax  :DATA:PATTern:PACKed <packing>
         :DATA:PATTern:PACKed?
Input   <packing>: Specify number of bits (1|4|8) that are encoded in a byte when
Parameters using the :DATA:PATT:IDAT command or query.
Return Range  1|4|8
Description This command specifies how many bits are encoded in a byte when using
the :DATA:PATT:IDAT command or query. The default is "8" for most
efficient coding.
Example  :DATA:PATTern:PACKed 1

:DATA:PATTern:IMPort
Syntax  :DATA:PATTern:IMPort <pattern-name>, <file-name>
Input   <pattern-name>: Specify the pattern name.
Parameters <file-name>: Specify the location of the pattern file to be imported.
Description This command imports an N4903B J-BERT pattern or M8020A/M8030A
pattern file.

M8020A/M8030A File Format:
File content consists of a short header and the actual symbol sequence.
The file content looks like the following:
Version=<version>
Use=<num-symbols>,<symbol-coding>[,<mask-used>[,<squelch-used>,<error-used>]]
[Description=<description>]
[Pack=<packing>]
Data=
<data>
Optional lines or fields are enclosed in square brackets. Format and
notation is intended to follow the concepts of the other :DATA:PATTern
SCPI commands. The first line must include a line with the file format
version. Accepted value for <version> is M8000 1.0.0 or M8000 1.0.1.
<error-used> field requires version 1.0.1.

Note that if <error-used> is 1, <mask-used> and <squelch-used> must both be 0. Also the coding must be BIT or B8B10. B128B130 and B128B132 are not supported.

The next line describes the number of symbols, symbol coding, mask usage and squelch usage as described in :DATA:PATTern:USE command.

The next line allows specifying a description.

The next line allows changing the packing of the actual <data> part similar to the :DATA:PATTern:PACK command. If not specified, 8 is assumed. For details on packing and its parameters, refer to :DATA:PATTern:PACKed[?] on page 397.

The <data> in a separate line contains the sequence of bytes as specified by the packing. For more details on how to edit data, refer to the chapter “Setting up Patterns” of the M8000 Series User Guide.

Example File:

Version=M8000 1.0.0
Use=4
Pack=1
Data=
1010

N4903B J-BERT Patterns

N4903B J-BERT patterns are always imported as bit coded patterns with no mask and no squelch. The B part of an alternate pattern will be imported as <pattern-name>B.

The following is an example of an N4903B J-BERT pattern import:

Example DATA:PATT:IMP "current:x","C:\N4903B\Pattern\xxx.ptrn"
:DATA:PATTern:EXPort

Syntax  
:DATA:PATTern:EXPort <pattern-name>, <file-name>

Input  

<pattern-name>: Specify the pattern name.

Parameters  
<file-name>: Specify the location of the pattern file to be exported.

Description  
This command exports a pattern to a file in the M8020A/M8030A import file format.

Example  
:DATA:PATT:EXP "current:x", "C:/Users/jsmith/Desktop/test"
:DATA:SEQuence Subnode

With this subsystem sequencing can be changed programmatically. For additional information, refer to Creating Pattern Sequences on page 43.

This subnode has the following SCPI structure:

```
:DATA
  SEQuence
    [:SET][?]
    [:SET]:VALue[?
    :NEW
    :NAMes?
    :BIND[?]
    :DELeTe
    :DALL
    :BREak
    :RESTart
    :IDENtifier
    :BLOCK?
    :GRANularity?
    :IDENtifier?
```

This subnode has the following commands:

Table 79

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:SET][?]</td>
<td>:DATA:SEQUence[:SET][?] on page 401</td>
</tr>
<tr>
<td>[:SET]:VALue[?]</td>
<td>:DATA:SEQUence[:SET]:VALue[?] on page 403</td>
</tr>
<tr>
<td>:NEW</td>
<td>:DATA:SEQUence:NEW on page 404</td>
</tr>
<tr>
<td>:BIND[?]</td>
<td>:DATA:SEQUence:BIND[?] on page 405</td>
</tr>
<tr>
<td>:DELeTe</td>
<td>:DATA:SEQUence:DELeTe on page 405</td>
</tr>
<tr>
<td>:DALL</td>
<td>:DATA:SEQUence:DALL on page 405</td>
</tr>
<tr>
<td>BREak</td>
<td>:DATA:SEQUence:BREak on page 406</td>
</tr>
</tbody>
</table>
:DATA:SEQUence[:SET]?

Syntax: :DATA:SEQUence[:SET] <sequence-name | location | group-name>, <pattern type>, <pattern-value>

:DATA:SEQUence[:SET]? <sequence-name | location>

Input Parameters:

- <sequence-name | location | group-name> is a string name for which possible values are:
  - <sequence-name>: Name of the existing sequence, either default sequence name or user created sequence name.
  - <location>: Individual Analyzer or Generator location e.g. "M1.DataIn1"
  - <group-name>: A user created group name containing Analyzer(s) or Generator(s) locations. In this case the group name must only contain numbers and letters.

- <pattern-type>
  - PRBS: Sets a PRBS of specified length.
  - PULSE: Sets a pulse of specified width.
  - CLOCK: Sets a clock of specified divider.
  - STATIC: Sets a static signal of specified value.
  - MEMORY: Sets a memory pattern saved in the setting.
<pattern-value>: String value parameter according to the pattern type.
  - For PRBS possible values are: "$2^{7}-1", "2^{9}-1", "2^{10}-1", "2^{11}-1", "2^{13}-1", "2^{15}-1", "2^{23}-1", "2^{23}-1p", "2^{31}-1", "2^{33}-1", "2^{35}-1", "2^{39}-1", "2^{41}-1", "2^{45}-1", "2^{47}-1", "2^{49}-1", "2^{51}-1"
  - For PULSE it’s a specified width value e.g. "$32"
  - For CLOCK it’s a specified divider value e.g. "$2"
  - For STATIC it’s a signal value e.g. "$0", "$1", "low", "high"
  - For MEMORY it’s a name of the pattern. Name consists of root node ("factory:", "shared:" or "current:" plus folder names separated by "/") plus pattern name. e.g. "factory:CEI/CEIstress_bit"

Description
This command allows you to:
  - set sequence for a specified location with the specified pattern.
  - set or change a pattern for single location, sequence or multiple locations (exiting user group).
  - create/modify a sequence for the given location with a single block pattern in an infinite loop.

For more verbose setting use, :DATA:SEQ:ENCE[SET]:VALue command providing XML value.

This query returns the following:
  - current pattern set for the given existing sequence or Analyzer/Generator location.
  - pattern and value if the current sequence for the given location contains single data block pattern.

If there are more than one block or block targets are used in the sequence then use, :Data:SEQ:SET:VALue? query to retrieve xml sequence setting value.

Example
The following are few command examples:

:DATA:SEQ:SET "Generator",PRBS,"2^{9}-1" sets a pattern for given "Generator" sequence.


The following are few query examples:

:DATA:SEQ:SET? "Generator"
> "Generator",PRBS,"2^9-1"

:DATA:SEQ:SET? "M1.DataIn1"
> "M1.DataIn1",PULSE,"32"

> "M1.DataOut1",CLOCK,"2^9-1"

:DATA:SEQ:SET? "Analyzer"
> "Analyzer",STATIC,"1"

:DATA:SEQ:SET? "M1.DataIn1"
> "M1.DataIn1",MEMORY,"factory:CEI/CEIstress_bit"

:DATA:SEQUence[:SET]:VALue[?] Syntax :DATA:SEQUence[:SET]:VALue <sequence-name>,<sequence-string>
DATA:SEQUence[:SET]:VALue? <sequence-name>

Input Parameters
<sequence-name>: Specify sequence name.
<sequence-string>: Enter definite length block and pattern sequence XML string.

Return Range <sequence-string>

Description This command is used to enter a sequence string consisting of the pattern sequence parameters for the specified sequence name and download it to the hardware. A definite length block must be entered indicating the length of the string to download.

**NOTE**

It is recommended to edit the sequence string in the Sequence Editor in either "UI" or "<Xml>" mode and then copy/paste the xml sequence string.
In the following example, #3340 defines the length of the string:

#3 indicates that a definite length arbitrary data block follows and its length is specified in 3 digits.

340 indicates that 340 bytes follow. In this example, the bytes are all readable ascii characters.

Example:
```
:DATA:SEQuence:VALue 'Generator',#3342<?xml version="1.0" encoding="utf-16"?>
<sequenceDefinition
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://www.keysight.com/schemas/M8000/DataSequence"><des
 cription />
<sequence>
<loop>
<block length="128"><prbs
 polynomial="2^7-1"
 /></block>
</loop></sequence></sequenceDefinition>
```

:DATA:SEQuence:NEW

Syntax: :DATA:SEQuence:NEW <sequence-name>

Input:  

Parameters:

Description:

This command is used to create a new sequence.

Example:
```
:DATA:SEQ:NEW 'MySequence'
```

:DATA:SEQuence:NAMes?

Syntax: :DATA:SEQuence:NAMes?

Description: This command returns the sequence names currently in use.

Example:
```
:DATA:SEQ:NAM?
"Generator","Analyzer","MySequence"
```
:DATA:SEQuence:BIND[?]

Syntax  
:DATA:SEQuence:BIND <sequence-name>,<identifier-list>

:DATA:SEQuence:BIND? <sequence-name>

Input
<sequence-name>: Specify sequence name.

Parameters
<identifier-list>: Specify location or group name identifier.

Return Range
<identifier-list>

Description
This command binds the identifiers to a specified sequence. The identifier is either a location or a group name identifier. If locations are already used in another sequence, they get re-assigned to this sequence.

This SCPI is applicable for M8041A, M8051A, M8195A, M8196A and M8046A. However, for M8195A and M8196A, it is only valid for Data Out locations.

Example  
:DATA:SEQ:BIND 'MySequence','M1.DataIn1'

:DATA:SEQuence:DELete

Syntax  
:DATA:SEQuence:DELete <sequence-name>

Input
<sequence-name>: Specify sequence name.

Parameters

Description
This command deletes the specified sequence.

Example  
:DATA:SEQ:DEL 'MySequence'

:DATA:SEQuence:DALL

Syntax  
:DATA:SEQuence:DALL

Description
This command deletes all sequences. This is a useful function if you do not want to use the reset sequence ("Generator", "Analyzer").

Example  
:DATA:SEQ:DALL
:**DATA:SEQ:**

**Sequence: BREak**

**Syntax:** :DATA:SEQ: BREAK 'identifier'

**Input:** *<identifier>*: 'M*.System'

**Parameters**

**Description**

This command sends a break signal addressed by the identifier. This will leave a loop and execute the next block or execute an arbitrary jump to another block.

**Example**

:DATA:SEQ: BRE 'M1.System'

---

**Sequence: REStart**

**Syntax:** :DATA:SEQ: REStart [<sequence-name>,<sequence-block-id>]

**Input:**

*<sequence-name>*: Specify sequence name.

*<sequence-block-id>*: Specify sequence block ID.

**Parameters**

**Description**

This command restarts all sequences if no argument is specified.

This command restarts a specified sequence if *<sequence-name>* is specified.

This command jumps to a specific sequence block if *<sequence-name>* and *<sequence-block-id>* is specified.

- Limitations on sequence block ID selection:
  - In case the sequencer block ID selected is contained inside an outer loop and it is not the starting segment the outer loop will not be executed.

**Example**

:DATA:SEQ: REST

:DATA:SEQ: REST 'Analyzer'

:DATA:SEQ: REST 'Generator', 3

---

**NOTE**

Analyzer(s) are not able to do a break. So using this command, you can branch to any other block, also on analyzers.
:DATA:SEQuence:RESTart:IDENtifier

**Syntax**
:DATA:SEQuence:RESTart:IDENtifier <identifier>

**Input**
<identifier>: 'M*.DataIn1', 'M*.DataIn2', 'M*.DataIn', 'M*.DataOut', 'M*.DataOut1' and 'M*.DataOut2'

**Parameters**
'M*.DataOut1' and 'M*.DataOut2'

**Description**
This command restarts the specified sequence (jumps to the first block). For example, this command is useful for ED sequences. Usually there is a synch block first and then a loop to compare the synchronized pattern. Performing a restart there causes a synchronization of the sequence.

**Example**
:DATA:SEQ:REST:IDEN 'M1.DataIn1'

:DATA:SEQuence:BLOCk?

**Syntax**
:DATA:SEQuence:BLOCk? <sequence-name>

**Input**
<sequence-name>: Specify sequence name.

**Parameters**

**Description**
This command returns the block of sequences currently executed.

**Example**
:DATA:SEQ:BLOC? 'Generator'

:DATA:SEQuence:BLOCk:IDENtifier?

**Syntax**
:DATA:SEQuence:BLOCk:IDENtifier? <identifier>

**Input**
<identifier>: 'M*.DataIn1', 'M*.DataIn2', 'M*.DataIn', 'M*.DataOut', 'M*.DataOut1' and 'M*.DataOut2'

**Parameters**
'M*.DataOut1' and 'M*.DataOut2'

**Description**
This command returns the block of sequences currently executed for a specific location.

**Example**
:DATA:SEQ:BLOC:IDEN? 'M1.DataIn1'
:DATA:SEQuence:GRANularity?

Syntax :DATA:SEQuence:GRANularity? <sequence-name>

Input :<sequence-name>: Specify sequence name, i.e., 'Sequence1'

Description This query returns the granularity in bits of sequence. It changes with the symbol width. The following table shows the granularity for various symbol widths for different modules:

<table>
<thead>
<tr>
<th>Symbol Width</th>
<th>Granularity (M8041A, M8051A, M8195A, M8196A)</th>
<th>Granularity (M8061A, M8082A)</th>
<th>Granularity (M8045A, M8046A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
<td>128</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>80</td>
<td>160</td>
<td>Not supported</td>
</tr>
<tr>
<td>130</td>
<td>130</td>
<td>260</td>
<td>Not supported</td>
</tr>
<tr>
<td>132</td>
<td>132</td>
<td>264</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Example :DATA:SEQ:GRAN? 'Sequence1'

:DATA:SEQuence:GRANularity:IDENtifier?

Syntax :DATA:SEQuence:GRANularity:IDENtifier? <identifier>

Input :<identifier>: 'M*.DataIn1', 'M*.DataIn2', 'M*.DataIn', 'M*.DataOut', 'M*.DataOut1' and 'M*.DataOut2'

Description This query returns the current granularity in bits of location or group. It changes with the symbol width. Refer to the table shown in the command :DATA:SEQuence:GRANularity:IDENtifier? on page 408.

Example :DATA:SEQ:GRAN:IDEN? 'M1.DataIn1'
This subnode has the following SCPI structure:

![](image)

This subnode has the following commands:

Table 80

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ONCE</td>
<td>:DATA:SYNC[:ONCE] on page 409</td>
</tr>
<tr>
<td>:TYPE[?]</td>
<td>:DATA:SYNC:TYPE[?] on page 410</td>
</tr>
<tr>
<td>:THReshold[?]</td>
<td>:DATA:SYNC:THReshold[?] on page 410</td>
</tr>
</tbody>
</table>

:DATA:SYNC[:ONCE]

**Syntax**  
:DATA:SYNC[:ONCE] <identifier>

**Input Parameters**  
<identifier>: 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'

**Description**  
This command initiates a pattern resynchronization.

This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

**Example**  
:DATA:SYNC 'M1.DataIn1'
### :DATA:SYNC:TYPE(?)

**Syntax**
:DATA:SYNC:TYPE 'identifier', AUTO|MANual
:DATA:SYNC:TYPE? 'identifier'

**Input Parameters**
- `<identifier>`: 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn'
- AUTO|MANual: Enable/disable automatic resynchronization.

**Return Range**
AUTO|MAN

**Description**
This command enables/disables automatic resynchronization. This SCPI is applicable for M8041A, M8051A, M8062A and M8046A.

**Example**
:DATA:SYNC:TYPE 'M1.DataIn1', AUTO

### :DATA:SYNC:THReshold(?)

**Syntax**
:DATA:SYNC:THReshold 'identifier', <NRf>
:DATA:SYNC:THReshold? 'identifier'

**Input Parameters**
- `<identifier>`: 'M*.DataIn1', 'M*.DataIn2' or 'M*DataIn', 'INF*.DataIn1'
- `<NRf>`: Set the threshold level.

**Return Range**
1E-8 to 1E-1

**Description**
This command sets the threshold level of error ratio at which synchronization is successful. Exceeding this BER threshold triggers a pattern synchronization when in automatic sync mode.

This SCPI is applicable for M8041A, M8051A, M8062A, M8046A and real-time oscilloscope based error detector channels.

**Example**
:DATA:SYNC:THR 'M1.DataIn1', 1E-5
:DATA:LINecoding Subnode

The :DATA:LINecoding subnode controls which line coding is being used when sending or receiving the digital data.

This subnode has the following SCPI structure:

```
:DATA
 :LINecoding
   [:VALue][?]
   :PAM3
     [:SYMBol]
     :LEVel(1)[?]
   :PAM4
     :MAPPing[?]
     :MAPPing:CUSTom[?]
     [:SYMBol]:LEVel(1|2)[?]
```

This subnode has the following commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DATA:LINecoding[:VALue][?]</td>
<td>:DATA:LINecoding[:VALue][?] on page 412</td>
</tr>
<tr>
<td>:DATA:LINecoding:PAM3[:SYMBol]:LEVel(1)[?]</td>
<td>:DATA:LINecoding:PAM3[:SYMBol]:LEVel(1)[?] on page 412</td>
</tr>
<tr>
<td>:DATA:LINecoding:PAM4:MAPPing[?]</td>
<td>:DATA:LINecoding:PAM4:MAPPing[?] on page 413</td>
</tr>
<tr>
<td>:DATA:LINecoding:PAM4[:SYMBol]:LEVel(1</td>
<td>2)[?]</td>
</tr>
</tbody>
</table>
:DATA:LINecoding[:VALue][?]  
Syntax: :DATA:LINecoding[:VALue] 'identifier', NRZ | PAM3 | PAM4  
:DATA:LINecoding[:VALue]? 'identifier'  
Input:  
Parameters:  
Description: The command selects the line coding to be used. The availability of individual selections depends on the hardware capabilities.  
PAM3 is only supported by M8045A.  
PAM4 is supported by M8045A, M8046A, M8195A, M8196A  
This command is not supported by instruments that only support NRZ, like M8041A, M8051A, M8061A, M8062A.  
This query returns the present state.  
This SCPI is applicable for M8045A.  
Example: :DATA:LIN 'M1.DataOut1', PAM4  
:DATA:LIN 'M2.DataIn', PAM4  

:DATA:LINecoding:PAM3[:SYMBol][:LEVel(1)][?]  
Syntax: :DATA:LINecoding:PAM3[:SYMBol][:LEVel(1)] <identifier>, <NRf>  
:DATA:LINecoding:PAM3[:SYMBol][:LEVel(1)]? 'identifier'  
Input: <identifier>: Any valid data output identifier that supports PAM3 encoding.  
Parameters: M*.DataOut1, M*.DataOut2  
Description: The command controls the PAM-3 symbol level 1. It can be used to adjust the actual output level that is being generated by a data output for PAM-3 symbol 1.  
The levels of PAM-3 symbol 0 and 2 cannot be adjusted and are always kept at the value of 0% and 100% of the actual output voltage swing.  
This query returns the present state.  
This SCPI is applicable for M8045A.  
**:DATA:LINiecoding:PAM4:MAPPing[?]**

**Syntax**

`:DATA:LINiecoding:PAM4:MAPPing 'identifier', NONE | GRAY | CUSTom

`:DATA:LINiecoding:PAM4:MAPPing? 'identifier'`

**Input**

<identifier>: Any valid data output or data input identifier. M*.DataOut1, M*.DataOut2, M*.DataOut3, M*.DataOut4, M*.DataIn and INF*.DataIn1

**Parameters**

M*.DataOut2, M*.DataOut3, M*.DataOut4, M*.DataIn and INF*.DataIn1

**Description**

The command selects how the pattern bits are mapped to symbol numbers.

- Select NONE if the mapping shall be ‘transparent’. This is useful if the pattern does already contain the required coding scheme. This option is available for M8195A, M8045A and M8046A.
- Select GRAY if consecutive pairs of bits shall be Gray coded in order to map to the PAM4 symbols. This option is available for M8195A M8045A and M8046A.
- Select CUSTom to use a user defined mapping definition. This option is only available for M8045A and M8046A.

This query returns the present state.

This SCPI is applicable for M8195A, M8196A, M8045A, M8046A and real-time oscilloscope based error detector channels.

**Example**

`:DATA:LIN:PAM4:MAPP 'M1.DataOut1', NONE

`:DATA:LIN:PAM4:MAPP 'M2.DataIn', CUST"

**:DATA:LINiecoding:PAM4:MAPPing:CUSTom[?]**

**Syntax**

`:DATA:LINiecoding:PAM4:MAPPing:CUSTom 'identifier', <"mapping definition">`

`:DATA:LINiecoding:PAM4:MAPPing:CUSTom? 'identifier'

**Input**

<identifier>: Any valid data output or data input identifier.

**Parameters**

**Description**

The command defines a custom bit sequence to symbol number mapping. The mapping defined with this command will be used when CUSTom is selected as the active mapping.

The mapping is defined as a string that contains a comma separated list of bit pairs. The position in the list corresponds to the symbol number.

"ww,xx,yy,zz" is mapping to Symbol 0 to Symbol 3 according to the following table.
This query returns the present state.
This SCPI is applicable for M8045A, M8046A, DSO634A, DSA634A and real-time oscilloscope based error detector channels.

Example: `DATA:LIN:PAM4:MAPP 'M1.DataIn1', '11,00,01,10'`

This command defines that Symbol 0 is being used for '11', Symbol 1 is being used for '00', Symbol 2 is being used for '01' and Symbol 3 is being used for '10'.

### Syntax

`:DATA:LINecoding:PAM4[:SYMBol]:LEVel(1|2)?`  
`:DATA:LINecoding:PAM4[:SYMBol]:LEVel(1|2) 'identifier', <NRf>`  
`:DATA:LINecoding:PAM4[:SYMBol]:LEVel(1|2)? 'identifier'`

### Input Parameters

- `<identifier>`: Any valid data output identifier. M*.DataOut1, M*.DataOut2, M*.DataOut3 and M*.DataOut4

### Range

- Symbol 1 Level: 1% to 65%
- Symbol 2 Level: 34% to 99%

### Description

The command controls the PAM4 level mappings. It can be used to adjust the actual output level that is being generated by a data output for a specific PAM4 symbol.

The levels of PAM4 symbol 0 and 3 cannot be adjusted and are always kept at the value of 0% and 100% of the actual output voltage swing.

It is best practice to configure symbol level 1 and 2 in a combined remote command transaction (see example below) to minimize hardware reconfiguration duration. This is especially important when the location is addressing a data output that is located on an arbitrary waveform generator module (e.g. M8195A).

This query returns the present state.

<table>
<thead>
<tr>
<th>Symbol Number</th>
<th>Bit Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol 0</td>
<td>ww</td>
</tr>
<tr>
<td>Symbol 1</td>
<td>xx</td>
</tr>
<tr>
<td>Symbol 2</td>
<td>yy</td>
</tr>
<tr>
<td>Symbol 3</td>
<td>zz</td>
</tr>
</tbody>
</table>
This SCPI is applicable for M8195A, M8196A and M8045A.

Example: 
```
:DATA:LIN:PAM4:LEV1 'M1.DataOut1', 35pct;LEVel2 'M1.DataOut1', 70pct
```

**:DATA:LINecoding:PAM4:PRECoding(?)**

**Syntax:** 
```
:DATA:LINecoding:PAM4:PRECoding 'identifier', <ON|OFF|1|0>
:DATA:LINecoding:PAM4:PRECoding? 'identifier'
```

**Input Parameters:**
- `<identifier>`: Any valid data output or data input identifier. M*.DataOut1, M*.DataOut2

**Description:**
The command enables/disables the pre-coding of the PAM4 symbols.

Pre-coding is an optional extra step between the symbol mapping and PAM4 level mapping. It is normally used together with a Gray code for the symbol mapping.

The pre-coder does not enforce using a Gray code for symbol mapping. This allows device specific custom symbol mapping (in case of LSB first devices), or using patterns that already contain the required symbol mapping (e.g. already gray encoded data).

This query returns the present state.

This SCPI is applicable for M8045A.

Example: 
```
:DATA:LIN:PAM4:PRE 'M1.DataOut1', ON
```

**:DATA:FEC Subnode**

The DATA:FEC subnode describes programming specification for the FEC error insertion feature.

FEC support in M8070B is enabled by installing a module specific FEC license. FEC licenses are available for M8045A.

The following FEC types are available:
- For IEEE 802.3cd 50GBASE-R
  - 1 lane at 26.5125GBd using PAM4
  - FEC RS(544,514)

FEC is enabled by selecting one of the FEC factory patterns:
- IEEE_802_3cd_RS_544_514_Scrambled_Idle
Due to the differences between FEC frame length and sequencer word width, there will be phases of invalid FEC data when transitioning from a FEC encoded sequence lock into a non-FEC sequence block and vice versa.

**:DATA:FEC:EINSertion Subsystem**

When a FEC pattern is used in the sequence, then the pattern generator offers a FEC specific error insertion.

The FEC error insertion is independent from the error insertion controlled with the **:OUTPut:EINSertion** subnode (see :OUTPut:EINSertion Subnode on page 233). Both error insertions can be used at the same time.

This subnode has the following SCPI structure:

![SCPI structure](image)

This subnode has the following commands:

**Table 82**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description under</th>
</tr>
</thead>
</table>
:DATA:FEC:EINSertion:STATe[?]

Syntax: :DATA:FEC:EINSertion[:STATe] 'identifier', <ON|OFF|1|0>
:DATA:FEC:EINSertion[:STATe]? 'identifier'

Input Parameters:
- <identifier>: Any valid data output identifier, 'M*.DataOut1','M*.DataOut2'

Return Range: 1 | 0

Description: This command enables or disables the continuous FEC error insertion. The inserted errors are after the FEC encoding.

This query returns the present state.

This SCPI is applicable for M8045A.

Example:
:DATA:FEC:EINS:STAT 'M1.DataOut1',ON
1

:DATA:FEC:EINSertion:ERRors[?]

:DATA:FEC:EINSertion:ERRors? 'identifier'

Input Parameters:
- <identifier>: Any valid data output identifier, 'M*.DataOut1','M*.DataOut2'

Return Range: 0 to 16

Description: This command specifies the number of symbols to be corrupted within a single FEC frame. The inserted errors are at random symbol positions. Each corrupted symbol will contain exactly one bit error at a random bit position.

This query returns the present state.

This SCPI is applicable for M8045A.

Example:
:DATA:FEC:EINS:ERR 'M1.DataOut1',8
8
### :DATA:FEC:EINSertion:BIP:ONCE[?]

**Syntax**

:DATA:FEC:EINSertion:BIP:ONCE 'identifier'  

**Input Parameters**

- `<identifier>`: Any valid data output identifier. 'M*.DataOut1','M*.DataOut2'

**Return Range**

1 | 0

**Description**

This command inserts a single Bit Interleaved Parity (BIP) error. This is a pre-FEC error and can be inserted at any time, even if the FEF error insertion is disabled.

This query returns the present state.

This SCPI is applicable for M8045A.

**Example**


Index

Symbols
*CLS, 102  
*ESE[?], 102  
*ESR?, 103  
*OPC, 89, 103  
*OPC?, 90, 103  
*OPT?, 104  
*RCL, 104  
*RST, 104  
*SAC, 104  
*SRE[?], 105  
*STB?, 105  
*TST?, 105  
*WAI, 89, 106

A
analyzer sequence, 44

B
block controls, global, 52  
block length, 48  
blocks, 47  
branch, conditional, 50

C
Clock Commands  
:CLKC[SOURce][?], 298, 300, 305

coding configuration  
MIPI Phy, 57  
PCIe 1/2, 56  
SAS, 57  
SATA, 56

USB 3.0, 55  
condition register, 29

Configuration  
32G BERT, 16, 17  
Base, 15, 19, 20  
M8020A 32 Gb/s BERT, 16, 17  
M8030A multi-channel BERT, 17, 19  
M8041A with M8061A, 16

D
Data Commands  
:DATA:PATTern:BLENgth?, 394  
:DATA:PATTern:CATalog?, 395  
:DATA:PATTern:CINFormation?, 3 95  
:DATA:PATTern:DELeate, 396  
:DATA:PATTern:DESCription[?], 3 96  
:DATA:PATTern:EXPort, 399  
:DATA:PATTern:IDATA[?], 394  
:DATA:PATTern:IMPORT, 397  
:DATA:PATTern:PACKed[?], 397  
:DATA:PATTern:USE[?], 393  
:DATA:SEQUence:BIND[?], 405  
:DATA:SEQUence:BLOCk:IDENTifier?, 407, 408  
:DATA:SEQUence:BLOCk?, 407  
:DATA:SEQUence:BREak, 406  
:DATA:SEQUence:DALL, 405  
:DATA:SEQUence:DELeate, 405  
:DATA:SEQUence:NAMEs?, 404  
:DATA:SEQUence:NEW, 402, 404  
:DATA:SEQUence:RESTart, 406  
:DATA:SEQUence:RESTart:IDENTifier, 407  
:DATA:SEQUence:VALue[?], 403  
:DATA:SYNC:THReshold[?], 259, 410, 412, 413, 414, 415, 417, 418  
:DATA:SYNC:TYPE[?], 410  
:DATA:SYNC[:ONCE], 409

Data Formats  
block data, 91  
boolean, 90  
numeric, 90  
string, 90  
download sequence, 60

E
event enable register, 30  
event register, 30

F
Fetch Commands  
:FETCh:BCOunter?, 312  
:FETCh:IBERate?, 315  
FlexDCA, 303  
FlexDCA N1000-Series System Software, 303

G
generator sequence, 44  
granularity, pattern generation, 46  
group name identifier, 22

I
Identifiers  
Group Name, 21  
Location, 21  
Measurement Name, 21  
IEEE, 88  
IEEE Mandatory Commands  
*CLS, 102  
*ESE[?], 102  
*ESR?, 102  
*IDN?, 103
Index

*OPC, 103
*OPC?, 103
*OPT?, 104
*RCL, 104
*RST, 104
*SAV, 104
*SRE[?], 105
*STB?, 105
*TST?, 105
*WAI, 106

IEEE mandatory commands, 101, 108, 109

Input Commands
:INPut:ALIGNment:EYE:ABORt, 2
:INPut:ALIGNment:EYE:RESult:POLarity?, 130, 276, 277
:INPut:ALIGNment:EYE:RESult:WI:DETh?, 266
:INPut:ALIGNment:EYE:TCEN, 27
:INPut:ALIGNment:EYE:THRESHold[?], 274
:INPut:CDR[:AUTO][?], 264
:INPut:CDR:CTRL[?], 263
:INPut:CDR:FIRSt:LBANdwidth[?], 266
:INPut:CDR:FIRSt:TEDENsity[?], 266
:INPut:CDR:FIRSt:HEight[?], 265
:INPut:CDR:LORDer[?], 263
:INPut:CDR:OPTimize, 265
:INPut:CDR:RELOck, 265
:INPut:CDR:SECond:LBANdwidth[?], 267

L

Link Training
Configuration, 58
Detect State, 51
Error, 51
Target State, 51
location identifier, 21
loop nesting, 47
loops, 47

M

M8020A
Four channel configuration, 15
Modules, 19
M8030A, 17
10-channel system configuration, 17, 19
M8030A-BU1, 17
M8030A-BU2, 17
M8040A, 19
M8041A module, 19
M8045A Pattern Generator Module, 19
M8045A, M8046A and M8057A configuration, 20
M8046A Analyzer Module, 20
M8057A Remote Head, 20
M8061A 32 Gb/s multiplexer with de-emphasis, 16
M8192A multi-channel synchronization module, 18
M9505A AXIe chassis, 19
M9514A, 18
M9514A AXIe 14-slot chassis, 17
M9537A AXIe Embedded Controller, 19
mask, 36
measurement name identifier, 22
Memory Commands
:MME:ORY:WORKspace:IMPort[:USER], 321
:MME:ORY:WORKspace:IMPort[:USER]:CATalog?, 322
:MME:ORY:WORKspace:IMPort[:USER]:DElete, 322
:MME:ORY:WORKspace:SETTing:FACTory:RECall, 320
:MME:ORY:WORKspace:SETTing[:USER]:DElete, 319
:MME:ORY:WORKspace:SETTing[:USER]:RECall, 319
:MME:ORY:WORKspace:SETTing[:USER]:STORE, 319

N

N71000 Subsystem, 303
nesting, loop, 47

O

operation status register, 34
Output Commands
:OUTPut:COUPling[?], 216
:OUTPut:DATA:F2Jitter[?], 218, 219
Index

output level behavior, 27
output protection, 26
output level behavior, 27
output protection, 26

P

Plugin Capture Commands
:PLUGin:CCAPture:ACQuisition:ALOCation?, 365
:PLUGin:CCAPture:ACQuisition:D EPTh?, 365
:PLUGin:CCAPture:ACQuisition:S OURce?, 366
:PLUGin:CCAPture:FETCh:ERRor: COUNt?, 359
:PLUGin:CCAPture:FETCh:PATTer n:ERRor:SYMBol:FIRST?, 360
:PLUGin:CCAPture:FETCh:FATTer n[:DATA]:DEPTh?, 360
:PLUGin:CCAPture:FETCh:FATTer n[:DATA]:HOLDoFF?, 359
:PLUGin:CCAPture:FETCh:RUN[: STA tus]?, 361
:PLUGin:CCAPture:NEW, 354
:PLUGin:CCAPture:RESet, 355
:PLUGin:CCAPture:SAVE, 356
:PLUGin:CCAPture:SHOW:BIT:[P ATterm]:VIEW[:]?], 357
:PLUGin:CCAPture:SHOW:SYMB ol:[PATtern]:VIEW[:]?], 357
:PLUGin:CCAPture:START, 355
:PLUGin:CCAPture:STOP, 355
:PLUGin:CCAPture:TRIGger:IMMediate[:ONCE], 356

Plugin Commands
:PLUGin:CATalog?, 330

Plugin Compare Commands
:PLUGin:CCAPture:RUN:LOG?, 3 70
:PLUGin:CCAPture:RUN:MESSag e?, 369
:PLUGin:CCAPture:RUN:PROGres s?, 369
:PLUGin:CCAPture:RUN:STATus?, 370

Plugin Error Ratio Commands
:PLUGin:ERATio:ACQuisition:ALO Cation[], 335
:PLUGin:ERATio:ACQuisition:DUR ation[], 337
:PLUGin:ERATio:ACQuisition:HIST ory[], 337
:PLUGin:ERATio:ACQuisition:INTer val[], 338
:PLUGin:ERATio:ACQuisition:TERa tio[], 339
:PLUGin:ERATio:CATalog[], 344
:PLUGin:ERATio:DELete, 345
:PLUGin:ERATio:EVAluation:CERa tid[], 340
:PLUGin:ERATio:EVAluation:RVM ode[], 341
:PLUGin:ERATio:FETCh:DATA?, 34 1
:PLUGin:ERATio:FETCh:DUReation [:], 344
:PLUGin:ERATio:FETCh:PRFresult?, 344
:PLUGin:ERATio:FETCh:DUReation [], 343
:PLUGin:ERATio:NEW, 345
:PLUGin:ERATio:RESet, 346
Index

:PLUGinERATio:RUN:HIStory[:STATe], 349
:PLUGinERATio:RUN:LOG?, 349
:PLUGinERATio:RUN:MESsage?, 350
:PLUGinERATio:RUN:PROGress?, 351
:PLUGinERATio:RUN[:STATe], 349
:PLUGinERATio:RUN:LOG?, 349
:PLUGinERATio:RUN:MESSage?, 350
:PLUGinERATio:RUN:PROGress?, 351
:PLUGinERATio:RUN[:STATe]?, 348, 351
:PLUGinERATio:START, 346
:PLUGinERATio:STOP, 347
plugin interface, 23
protection algorithm, 26
Q
questionable status register, 34
R
registers, 28
S
safety summary, 3
SCPI
command keywords, 92
command separators, 93
command structure, 94
command syntax, 92
common commands, 88
concepts, 91
consortium, 88
instrument control commands, 88
instrument model, 91
mandatory commands, 89
overlapped and sequential commands, 89
query response, 93
send commands, 95, 96, 97
subsystems, 100
VISA, sending commands using, 95
SCPI Standard Commands
:SYStem:ERRor:NEXT?, 107
:SYStem:HELP:HEADers?, 107
:SYStem:VERsion?, 107
scrambler, 54
sequence editor, 43
sequence, download, 60
sequences, building, 45
sequences, pattern, 43
Source Commands
[:SOURce]:VOLTage:AMPLitude?, 109
[:SOURce]:CONFigure:MINTegration?, 207, 208, 210
[:SOURce]:FREQuency?, 141
[:SOURce]:INTerference:CMODE:AMPLitude?, 196
[:SOURce]:INTerference:CMODE:GAIN?, 197
[:SOURce]:INTerference:CMODE:SOURce?, 196
[:SOURce]:INTerference:CMODE[:STATe]?, 196
[:SOURce]:INTerference:DMODE:AMPLitude?, 198
[:SOURce]:INTerference:DMODE:GAIN?, 198, 199, 200
[:SOURce]:INTerference:DMODE:SOURce?, 198
[:SOURce]:INTerference:DMODE[:STATe]?, 197
[:SOURce]:INTerference:HFRequency[:SOURce]?, 195, 204, 205
[:SOURce]:INTerference:HFRequency:CONFigure[:DElay]?], 195
[:SOURce]:JITTer:HFRequency:BUNCorrelate:AMPLitude?, 16
[:SOURce]:JITTer:HFRequency:BUNCorrelate:DRAte?, 164
[:SOURce]:JITTer:HFRequency:BUNCorrelate:FILTER[:SE lect]?, 164, 165
[:SOURce]:JITTer:HFRequency:BUNCorrelate:STATe?], 163
[:SOURce]:JITTer:HFRequency:EXTernal[:STATe]?, 157
[:SOURce]:JITTer:HFRequency:ERiodic(*):AMPLitude[], 160
[:SOURce]:JITTer:HFRequency:ERiodic(*):FREQuency[], 161
[:SOURce]:JITTer:HFRequency:PRiodic(*)[:STATe]?, 160
[:SOURce]:JITTer:HFRequency:PRiodic(*)[:STATE]?, 167
[:SOURce]:JITTer:HFRequency:R ANDom:AMPLitude?, 158
[:SOURce]:JITTer:HFRequency:R ANDom:AMPLitude:, 167
[:SOURce]:JITTer:HFRequency:R ANDom:FILTer[:LPASs]?, 168
[:SOURce]:JITTer:HFRequency:R ANDom:FILTer[:LPASs]?, 169
[:SOURce]:JITTer:HFRequency:R ANDom[:STATe]?, 167
[:SOURce]:JITTer:HFRequency:SPEctrally:AMPLitude(*)?, 171
[:SOURce]:JITTer:HFRequency:SPEctrally:AMPLitude(*)?, 171
[:SOURce]:JITTer:HFRequency:SPEctrally[:STATe]?, 172
[:SOURce]:JITTer:HFRequency:U NITY?], 157
[:SOURce]:JITTer:LFRequency:PEridic:AMPLitude[], 176, 177
[:SOURce]:JITTer:LFRequency:PEridic:AMPLitude[], 176, 177
[:SOURce]:JITTer:LFRequency:PEridic:FREQuency[], 177, 178
[:SOURce]:JITTer:LFRequency:PEridic[:STATe]?, 176, 177
[:SOURce]:JITTer:LFRequency:R SSClocking:AMPLitude[], 180
[:SOURce]:JITTer:LFRequency:R SSClocking:AMPLitude[], 180
[:SOURce]:JITTer:LFRequency:R SSClocking:FREQuency[], 180
[:SOURce]:JITTer:LFRequency:R SSClocking[:STATE]?, 179

422 Keysight M8000 Series of BER Test Solutions Programming Guide