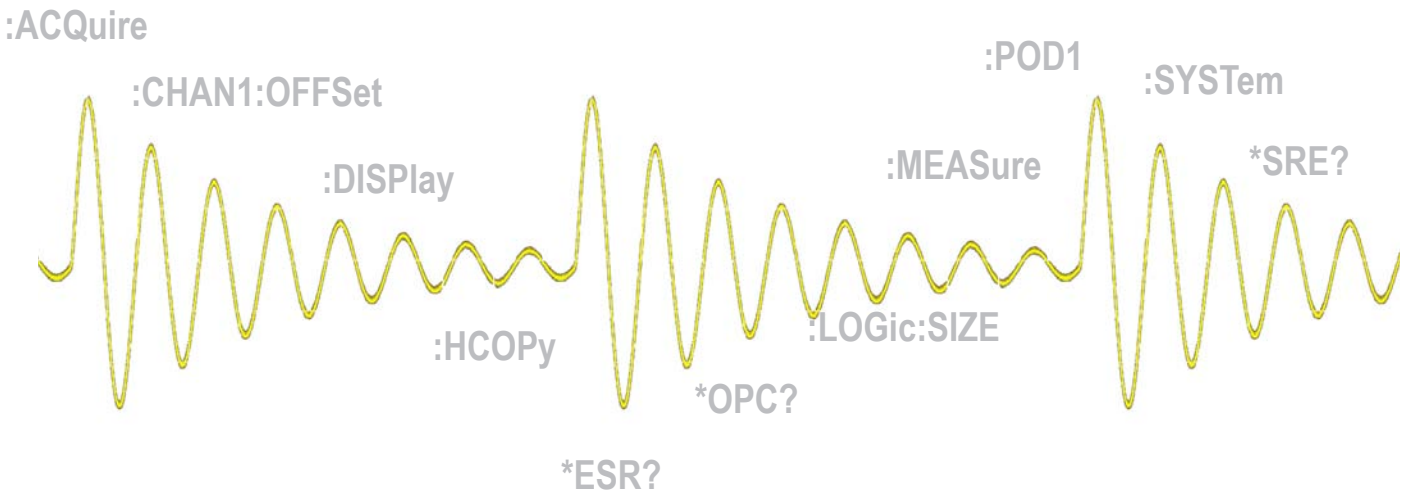
HAMEG Instruments
A Rohde&Schwarz Company

SCPI Programmers Manual
HMO Series

HMO352x, HMO2524: Firmware V 4.202
HMO72x ... HMO202x: Firmware V 4.522

English



HAMEG Instruments GmbH • Industriestr. 6 • D-63533 Mainhausen • Deutschland
Tel.: +49 (0) 6182 800 0 • Fax: +49 (0) 6182 800 100 • E-Mail: info@hameg.com
Geschäftsführer: Dipl.-Ing. Holger Asmussen, Dipl.-Ing. Andre Vander Stichelen • AG Offenbach am Main HRB 41200

Inhalt

1. Introduction / Basics

This chapter provides basic information on operating an instrument via remote control.

1.1 Remote Control Interfaces

For remote control, RS-232/USB (standard interface HO720) or LAN/USB (optional interface HO730) or GPIB (optional interface HO740) can be used. The optional interfaces replaces the RS-232/USB HO720 interface module on the rear panel.

Within this interface description, the term GPIB is used as a synonym for the IEC/IEEEbus interface.

SCPI (Standard Commands for Programmable Instruments) SCPI commands - messages - are used for remote control. Commands that are not taken from the SCPI standard follow the SCPI syntax rules.

1.1.1 RS-232

The oscilloscope is equipped with a RS-232/USB interface and can be connected to an computer using either classic RS-232 or USB.

If the computer does not have a RS-232 (COM) Port, please use USB or virtual COM Port via USB. Use of USB - RS-232 (serial) Converter may not work.

If you use classic RS-232 you do not need any driver. In order to set these parameter at the HMO, please press the button SETUP at the front panel in the area GENERAL and hit the soft key INTERFACE at the opened soft menu. Make sure the RS-232 interface is chosen (blue backlighted) and then hit the button PARAMETER. This opens a menu where you can set and save all parameter for the RS-232 communication. Setting of the RS-232 must fit the setting of the corresponding COM Port at the Computer.

The VISA resource string has the form: ASRL<n>::INSTR

<n>	COM-Port number
Example:	ASRL1::INSTR

1.1.2 USB Interface

If you are using USB you need to install a USB driver, which is available for download free of charge from the HAMEG Website.

All descriptions regarding the USB interface are true for the HO720 interface card as well as for the optional HO730 USB part. All currently available USB driver are fully tested, functional and released for Windows XP™ 32 Bit, Windows Vista™ or Windows 7™ both as 32Bit or 64Bit versions.

The USB interface must be chosen in the oscilloscope and does not need any setting. The connection can be done via the normal USB or via the virtual COM port.

If the virtual COM port will be used, you must set USB as interface at the oscilloscope, install the VCOM part of USB driver at the Computer.

The VISA resource string has the form:

USB::<manufacturer-ID>::<model code>::<IF-serial number>::RAW

1.1.3 LAN Interface

The HMO series can be equipped with a network interface and can be connected to an Ethernet LAN (local area network) for remote control of the instrument. The instrument accepts remote commands via the LAN interface using the VISA library. VISA must be installed on the control computer.

The HMO series supports two ways of LAN communication:

- **VXI-11 protocol**: a protocol that has been specifically developed for test and measurement instruments. It is the recommended protocol for remote control via LAN.

- **Raw socket mode**: a synchronous, streaming oriented protocol. Consequently, raw socket communication does not support asynchronous events like Service Request (SRQ) or Device Clear (DCL).

The settings of the parameter at the oscilloscope are done after selecting ETHERNET as the interface and the soft key PARAMETER is chosen. You can set anything including a fix IP adress. Alternative you can chose a dynamic IP setting via the DHCP function. Please ask your IT department for the correct setting at your network.

IP address
Only the IP address of the instrument is required to set up the connection. It is part of the resource string used by the programs to identify and control the instrument. The VISA resource string has the form:

TCPIP::<IP_address>::<IP_port>::SOCKET

The default port number for SCPI socket communication is 5025. IP address and port number are listed In the „Ethernet Settings" of the oscilloscope, see also: chapter 1.2.2,"Configuring LAN Parameters", on page 4.

Example: If the instrument has the IP address 192.1.2.3; the valid VISA resource string is:

TCPIP::192.1.2.3::5025::SOCKET

The end character must be set to linefeed.

VXi11-Protocol
The VXI-11 standard is based on the ONC RPC (Open Network Computing Remote Procedure Call) protocol which in turn relies on TCP/IP as the network/transport layer. The TCP/IP network protocol and the associated network services are preconfigured.

The valid VISA resource string is: TCPIP::<IP>::INSTR
<IP> IP-Address
Example: TCPIP::192.168.155.1::INSTR

1.1.4 GPIB Interface (IEC/IEEE Bus Interface)

To be able to control the instrument via the GPIB bus, the instrument and the controller must be linked by a GPIB bus cable. A GPIB bus card, the card drivers and the program libraries for the programming language must be provided in the controller. The controller must address the instrument with the GPIB instrument address.

Characteristics

The GPIB interface is described by the following characteristics:
- Up to 15 instruments can be connected
- The total cable length is restricted to a maximum of 15 m; the cable lenth between two instruments should not exceed 2 meters.
- A wired „OR"-connection is used if several instruments are connected in parallel.

GPIB Instrument Address

In order to operate the instrument via remote control, it must be addressed using the GPIB address. The remote control address is factory-set to 20, but it can be changed in the network environment settings or in the „Setup" menu under „Interface > Parameter". For remote control, addresses 0 through 30 are allowed. The GPIB address is maintained after a reset of the instrument settings.

The valid VISA resource string is: GPIB::<n>::INSTR
<n> GPIB address
Example: GPIB::1::INSTR

1.2 Setting Up a Network (LAN) Connection

1.2.1 Connecting the Instrument to the Network

The network card can be operated with a 10 Mbps Ethernet IEEE 802.3 or a 100 Mbps Ethernet IEEE 802.3u interface.

Risk of network failure
Before connecting the instrument to the network or configuring the network, consult your network administrator. Errors may affect the entire network.

To establish a network connection, connect a commercial RJ-45 cable to one of the LAN ports of the instrument and to a PC.

1.2.2 Configuring LAN Parameters

Depending on the network capacities, the TCP/IP address information for the instrument can be obtained in different ways.
- If the network supports dynamic TCP/IP configuration, using the Dynamic Host Configuration Protocol (DHCP), and a DHCP server is available, all address information can be assigned automatically.
- Otherwise, the address must be set manually. Automatic Private IP Addressing (APIPA) is not supported.

By default, the instrument is configured to use dynamic TCP/IP configuration and obtain all address information automatically. This means that it is safe to establish a physical connection to the LAN without any previous instrument configuration.

Connection errors can affect the entire network. If your network does not support DHCP, or if you choose to disable dynamic TCP/IP configuration, you must assign valid address information before connecting the instrument to the LAN. Contact your network administrator to obtain a valid IP address.

Configuring LAN parameters

1. Press the SETUP key and then the „INTERFACE" softkey.

2. Press the „ETHERNET" softkey and then the „PARAMETER" softkey.
   Note: By default, the instrument is set to DHCP. If the instrument is set to use DHCP and cannot find a DHCP server, it takes about two minutes until the ETHERNET menu is available.

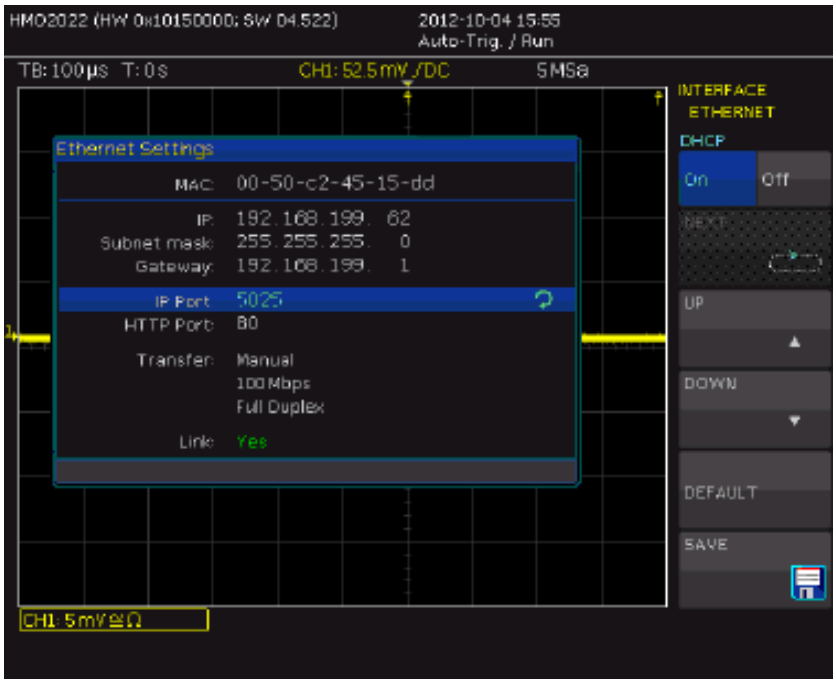


Fig. 1.1: Ethernet Settings dialog box

Some data is displayed for information only and cannot be edited. This includes the „MAC" (physical) address of the connector and the „Link" status information.

3. If the LAN does not support DHCP, or the instrument is directly connected with a computer, disable DHCP: Press the „DHCP" softkey so that „OFF" is highlighted.

4. Define the IP address of the instrument by entering each of the four blocks individually.

   a) Define the first block number using the Universal knob.
   b) Press „Next" to move to the next block and define the number.
   c) When the IP address is complete, press „Down" to continue with the next setting.

5. Define the „Subnetmask" and „Gateway" in the same way as the IP address.

6. Select the „IP Port" - the port number for SCPI socket communication.

7. Select the „HTTP Port" used by the instrument.

8. Select the „Transfer" mode. This mode can either be determined automatically („Auto" setting), or you can select a combination of a transfer rate and half or full duplex manually.

9. Press „Save" to save the LAN parameters on the instrument.

The „Link" status information at the bottom of the dialog box indicates whether a LAN connection was established successfully.

Checking LAN and SCPI connection

1. Check the LAN connection using ping: `ping xxx.yyy.zzz.xxx`.

2. If the computer can access the instrument, enter the IP address of the instrument in the address line of the internet browser on your computer: http//:xxx.yyy.zzz.xxx

   The „Instrument Home" page appears. It provides information on the instrument and the LAN connection.

Fig. 1.2: The „Instrument Home" page provides information on the instrument and the LAN connection.

3. On the „Screen Data" page, a copy of the instrument screen is shown. To get the current screen content from the instrument, click „Update".
   To save the screen copy, right-click the picture and select „Save picture as".

4. On the „SCPI Device Control" page, you can check if the transfer of remote commands is working. You can enter a single command, for example; ***IDN?**, and transmit it with „Send".

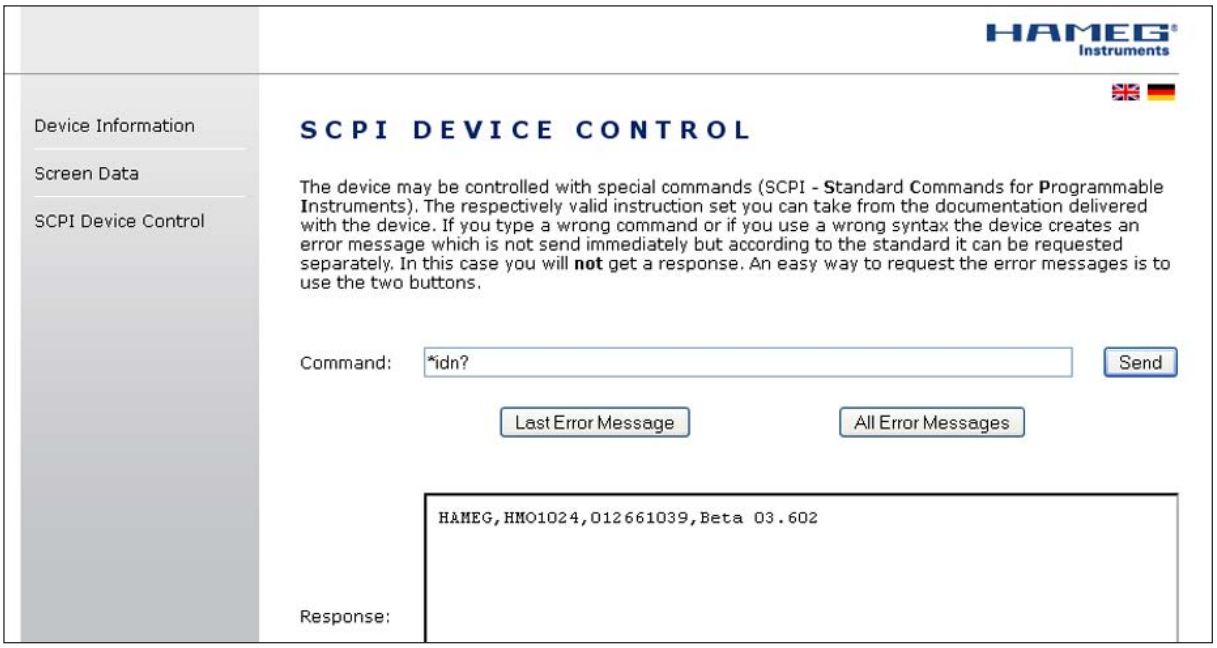


Fig. 1.3: The SCPI Device Control

1.3 Switching to Remote Control

When you switch on the instrument, it is always in manual operation state („local" state) and can be operated via the front panel.

When you send a command from the control computer, it is received and executed by the instrument. The display remains on, manual operation via the front panel is always possible.

The instrument front panel control will be activated once again by pushing a menu button or by sending a SCPI remote command. Otherwise, the front panel is locked.

The remote mode will be activated with:
REN = with GPIB

and with the SCPI command
&GTR. = Go To Remote

Return to the local mode:
&GTL. = Go To Local

1.4 Messages and Command Structure

1.4.1 Messages

Instrument messages are employed in the same way for all interfaces, if not indicated otherwise in the description.

See also:
– Structure and syntax of the instrument messages: chapter 1.4.2, „SCPI Command Structure", on page 9
– Detailed description of all messages: chapter 2, „Command Reference", on page 28

There are different types of instrument messages, depending on the direction they are sent:
– Commands
– Instrument responses

Commands

Commands (program messages) are messages the controller sends to the instrument. They operate the instrument functions and request information. The commands are subdivided according to two criteria:

1. **According to the effect they have on the instrument:**

   – **Setting commands** cause instrument settings such as a reset of the instrument or setting the frequency.

   – **Queries** cause data to be provided for remote control, e.g. for identification of the instrument or polling a parameter value. Queries are formed by directly appending a question mark to the command header.

2. **According to their definition in standards:**

   – **Common commands:** their function and syntax are precisely defined in standard IEEE 488.2. They are employed identically on all instruments (if implemented). They refer to functions such as management of the standardized status registers, reset and self test.

   – **Instrument control commands** refer to functions depending on the features of the instrument such as frequency settings. Many of these commands have also been standardized by the SCPI committee. These commands are marked as „SCPI compliant" in the command reference chapters. Commands without this SCPI label are device-specific, however, their syntax follows SCPI rules as permitted by the standard.

Instrument responses

Instrument responses (response messages and service requests) are messages the instrument sends to the controller after a query. They can contain measurement results, instrument settings and information on the instrument status.

Interface Messages

The interface messages are called low–level control messages. These messages can be used to emulate interface messages of the GPIB bus.

Table 1.1: Interface Messages

Command	Long term	Effect on the instrument
&DCL	Device Clear	Aborts processing of the commands just received and sets the command processing software to a defined initial state. Does not change the instrument setting.
&GTL	Go to Local	Transition to the „local“ state (manual control).
&GTR	Go to Remote	Transition to the „remote“ state (remote control).
&LLO	Local Lockout	Disables switchover from remote control to manual control by means of the front panel keys.
&NREN	Not Remote Enable	Enables switchover from remote control to manual operation by means of the front panel keys

GPIB Interface Messages

Interface messages are transmitted to the instrument on the data lines, with the attention line (ATN) being active (LOW). They are used for communication between the controller and the instrument and can only be sent by a computer which has the function of a GPIB bus controller. GPIB interface messages can be further subdivided into:

- **Universal commands:** act on all instruments connected to the GPIB bus without previous addressing
- **Addressed commands:** only act on instruments previously addressed as listeners

Universal Commands

Universal commands are encoded in the range 10 through 1F hex. They affect all instruments connected to the bus and do not require addressing.

Table 1.2: Universal Commands

Command	Effect on the instrument
DCL (Device Clear)	Aborts the processing of the commands just received and sets the command processing software to a defined initial state. Does not change the instrument settings.
IFC (Interface Clear) *)	Resets the interfaces to the default setting.
LLO (Local Lockout)	The LOC/IEC ADDR key is disabled.
SPE (Serial Poll Enable)	Ready for serial poll.
SPD (Serial Poll Disable)	End of serial poll.
PPU (Parallel Poll Unconfigure)	End of the parallel-poll state.
*) IFC is not a real universal command, it is sent via a separate line; however, it also affects all instruments connected to the bus and does not require addressing	

Addressed Commands

Addressed commands are encoded in the range 00 through 0F hex. They only affect instruments addressed as listeners.

Table 1.3: Addressed Commands

Command	Effect on the instrument
GET (Group Execute Trigger)	Triggers a previously active instrument function (e.g. a sweep). The effect of the command is the same as with that of a pulse at the external trigger signal input.
GTL (Go to Local)	Transition to the „local“ state (manual control).
GTR (Go to Remote)	Transition to the „remote“ state (remote control).
PPC (Parallel Poll Configure)	Configures the instrument for parallel poll.
SDC (Selected Device Clear)	Aborts the processing of the commands just received and sets the command processing software to a defined initial state. Does not change the instrument setting.

1.4.2 SCPI Command Structure

SCPI commands consist of a so-called header and, in most cases, one or more param-eters. The header and the parameters are separated by a „white space“ (ASCII code 0 to 9, 11 to 32 decimal, e.g. blank). The headers may consist of several mnemonics (keywords). Queries are formed by appending a question mark directly to the header.

The commands can be either device-specific or device-independent (common commands). Common and device-specific commands differ in their syntax.

Syntax for Common Commands

Common (=device-independent) commands consist of a header preceded by an asterisk (*) and possibly one or more parameters.

Examples:

Table 1.4: Examples of Common Commands

*RST	RESET	Resets the instrument.
*ESE	EVENT STATUS ENABLE	Sets the bits of the event status enable registers.
*ESR?	EVENT STATUS QUERY	Queries the contents of the event status register.
*IDN?	IDENTIFICATION QUERY	Queries the instrument identification string.

Syntax for Device-Specific Commands

Not all commands used in the following examples are necessarily implemented in the instrument.

For demonstration purposes only, assume the existence of the following commands for this section:

- `DISPlay[:WINDow<1...4>]:MAXimize <Boolean>`
- `HardCOPy:DEVice:COLor <Boolean>`
- `HardCOPy:DEVice:CMAP:COLor:RGB <red>,<green>,<blue>`
- `HardCOPy[:IMMediate]`

- `HardCOPy:ITEM:ALL`
- `HardCOPy:ITEM:LABel <string>`
- `HardCOPy:PAGE:DIMensions:QUADrant[<N>]`
- `HardCOPy:PAGE:ORIentation LANDscape | PORTrait`
- `HardCOPy:PAGE:SCALe <numeric value>`
- `MMEMory:COPY <file_source>,<file_destination>`

Long and short form

The mnemonics feature a long form and a short form. The short form is marked by upper case letters, the long form corresponds to the complete word. Either the short form or the long form can be entered; other abbreviations are not permitted.

Example:

`HardCOPy:DEVice:COLor ON` is equivalent to `HCOP:DEV:COL ON`.

Case-insensitivity
Upper case and lower case notation only serves to distinguish the two forms in the manual, the instrument itself is case-insensitive.

Numeric suffixes

If a command can be applied to multiple instances of an object, e.g. specific channels or sources, the required instances can be specified by a suffix added to the command.
Numeric suffixes are indicated by angular brackets (<1...4>, <n>, <i>) and are replaced by a single value in the command. Entries without a suffix are interpreted as having the suffix 1.

Example:

Definition: `HardCOPy:PAGE:DIMensions:QUADrant[<N>]`
Command: `HCOP:PAGE:DIM:QUAD2`
This command refers to the quadrant 2.

Different numbering in remote control
For remote control, the suffix may differ from the number of the corresponding selection used in manual operation. SCPI prescribes that suffix counting starts with 1. Suffix 1 is the default state and used when no specific suffix is specified.

Optional mnemonics

Some command systems permit certain mnemonics to be inserted into the header or omitted. These mnemonics are marked by square brackets in the description. The instrument must recognize the long command to comply with the SCPI standard. Some commands are considerably shortened by these optional mnemonics.

Example:

Definition: `HardCOPy[:IMMediate]`
Command: `HCOP:IMM` is equivalent to `HCOP`

Optional mnemonics with numeric suffixes
Do not omit an optional mnemonic if it includes a numeric suffix that is relevant for the effect of the command.

Example:

Definition: `DISPlay[:WINDow<1...4>]:MAXimize <Boolean>`
Command: `DISP:MAX ON` refers to window 1.

In order to refer to a window other than 1, you must include the optional WINDow param-eter with the suffix for the required window.

DISP:WIND2:MAX ON refers to window 2.

Parameters

Parameters must be separated from the header by a „white space". If several parameters are specified in a command, they are separated by a comma (,). For a description of the parameter types, refer to „SCPI Parameters", on page 12.

Example:

Definition: `HardCOPy:DEVice:CMAP:COLor:RGB <red>,<green>,<blue>`
Command: `HCOP:DEV:CMAP:COL:RGB 3,32,44`

Special characters

Table 1.5: Special characters

|	**Parameters**
	A vertical stroke in parameter definitions indicates alternative possibilities in the sense of „or". The effectof the command differs, depending on which parameter is used.
	Example:
	Definition: `HardCOPy:PAGE:ORIentation LANDscape | PORTrait`
	Command: `HCOP:PAGE:ORI LAND` specifies landscape orientation
	Command: `HCOP:PAGE:ORI PORT` specifies portrait orientation
	Mnemonics
	A selection of mnemonics with an identical effect exists for several commands. These mnemonics are indicated in the same line; they are separated by a vertical stroke. Only one of these mnemonics needs to be included in the header of the command. The effect of the command is independent of which of the mnemonics is used.
	Example:
	Definition `SENSE:BANDwidth|BWIDth[:RESolution] <numeric_value>`
	The two following commands with identical meaning can be created:
	`SENS:BAND:RES 1`
	`SENS:BWID:RES 1`

[ ]	mnemonics in square brackets are optional and may be inserted into the header or omitted. **Example:** `HardCOPy[:IMMediate]` `HCOP:IMM` is equivalent to `HCOP`

{ }	Parameters in curly brackets are optional and can be inserted once or several times, or omitted.
	Example: `SENSe:LIST:FREQuency <numeric_value>{,<numeric_value>}`
	The following are valid commands:
	`SENS:LIST:FREQ 10`
	`SENS:LIST:FREQ 10,20`
	`SENS:LIST:FREQ 10,20,30,40`

SCPI Parameters

Many commands are supplemented by a parameter or a list of parameters. The parameters must be separated from the header by a „white space" (ASCII code 0 to 9, 11 to 32 decimal, e.g. blank). Allowed parameters are:

- Numeric values
- Special numeric values
- Boolean parameters
- Text
- Character strings
- Block data

The parameters required for each command and the allowed range of values are specified in the command description.

Numeric values

Numeric values can be entered in any form, i.e. with sign, decimal point and exponent. Values exceeding the resolution of the instrument are rounded up or down. The mantissa may comprise up to 255 characters, the exponent must lie inside the value range -32000 to 32000. The exponent is introduced by an „E" or „e". Entry of the exponent alone is not allowed. In the case of physical quantities, the unit can be entered. Allowed unit prefixes are G (giga), MA (mega), MOHM and MHZ are also allowed), K (kilo), M (milli), U (micro) and N (nano). If the unit is missing, the basic unit is used.

Example: `SENSe:FREQ:STOP 1.5GHz = SENSe:FREQ:STOP 1.5E9`

Units

For physical quantities, the unit can be entered. Allowed unit prefixes are:
- G (giga)
- MA (mega), MOHM, MHZ
- K (kilo)
- M (milli)
- U (micro)
- N (nano)

If the unit is missing, the basic unit is used.

Example: `SENSe:FREQ:STOP 1.5GHz = SENSe:FREQ:STOP 1.5E9`

Some settings allow relative values to be stated in percent. According to SCPI, this unit is represented by the PCT string.

Example: `HCOP:PAGE:SCAL 90PCT`

Special numeric values

The texts listed below are interpreted as special numeric values. In the case of a query, the numeric value is provided.

- MIN/MAX
  MINimum and MAXimum denote the minimum and maximum value.

Example:

Setting command: `SENSe:LIST:FREQ MAXimum`
Query: `SENS:LIST:FREQ?, Response: 3.5E9`

Queries for special numeric values
The numeric values associated to MAXimum/MINimum can be queried by adding the corresponding mnemonics to the command. They must be entered following the quotation mark.

Example: `SENSe:LIST:FREQ? MAXimum`

Returns the maximum numeric value as a result.

Boolean Parameters

Boolean parameters represent two states. The „ON" state (logically true) is represented by „ON" or a numeric value 1. The „OFF" state (logically untrue) is represented by „OFF" or the numeric value 0. The numeric values are provided as the response for a query.

Example:

Setting command:	`HCOPy:DEV:COL ON`
Query:	`HCOPy:DEV:COL?`
Response:	`1`

Text parameters

Text parameters observe the syntactic rules for mnemonics, i.e. they can be entered using a short or long form. Like any parameter, they have to be separated from the header by a white space. In the case of a query, the short form of the text is provided.

Example:

Setting command:	`HardCOPy:PAGE:ORIentation LANDscape`
Query:	`HCOP:PAGE:ORI?`
Response:	`LAND`

Character strings

Strings must always be entered in quotation marks (' or „).

Example: `HCOP:ITEM:LABel „Test1"` or `HCOP:ITEM:LABel 'Test1'`

Block data

Block data is a format which is suitable for the transmission of large amounts of data. A command using a block data parameter has the following structure:

Example: `FORMat:READings:DATA #45168xxxxxxxx`

The ASCII character # introduces the data block. The next number indicates how many of the following digits describe the length of the data block. In the example the 4 following digits indicate the length to be 5168 bytes. The data bytes follow. During the transmission of these data bytes all end or other control signs are ignored until all bytes are transmitted. #0 specifies a data block of indefinite length. The use of the indefinite format requires a `NL^END` message to terminate the data block. This format is useful when the length of the transmission is not known or if speed or other considerations prevent segmentation of the data into blocks of definite length.

Overview of Syntax Elements

The following table provides an overview of the syntax elements:

Table 1.6: Syntax Elements

:	The colon separates the mnemonics of a command. In a command line the separating semicolonmarks the uppermost command level.
;	The semicolon separates two commands of a command line. It does not alter the path.
,	The comma separates several parameters of a command.
?	The question mark forms a query.
*	The asterisk marks a common command.
“	Quotation marks introduce a string and terminate it.
#	The hash symbol introduces binary, octal, hexadecimal and block data. – Binary: #B10110 – Octal: #O7612 – Hexa: #HF3A7 – Block: #21312
	A „white space“ (ASCII-Code 0 to 9, 11 to 32 decimal, e.g. blank) separates the header from the parameters.

Structure of a command line

A command line may consist of one or several commands. It is terminated by one of the following:
– a <New Line>
– a <New Line> with EOI
– an EOI together with the last data byte

Several commands in a command line must be separated by a semicolon „;“. If the next command belongs to a different command system, the semicolon is followed by a colon.

Example: `MMEM:COPY „Test1“,“MeasurementXY“;:HCOP:ITEM ALL`
This command line contains two commands. The first command belongs to the MMEM system, the second command belongs to the HCOP system.

If the successive commands belong to the same system, having one or several levels in common, the command line can be abbreviated. To this end, the second command after the semicolon starts with the level that lies below the common levels. The colon following the semicolon must be omitted in this case.

Example: `HCOP:ITEM ALL; HCOP:IMM`

This command line is represented in its full length and contains two commands separated from each other by the semicolon. Both commands are part of the HCOP command system, i.e. they have one level in common.When abbreviating the command line, the second command begins with the level below HCOP. The colon after the semicolon is omitted. The abbreviated form of the command line reads as follows:

```
HCOP:ITEM ALL;IMM
```

However, a new command line always begins with the complete path.

Example:
```
HCOP:ITEM ALL
HCOP:IMM
```

Responses to Queries

A query is defined for each setting command unless explicitly specified otherwise. It is formed by adding a question mark to the associated setting command. According to SCPI, the responses to queries are partly subject to stricter rules than in standard IEEE 488.2.

- The requested parameter is transmitted without a header.
  Example: `HCOP:PAGE:ORI?,`
  Response: `LAND`

- Maximum values, minimum values and all other quantities that are requested via a special text parameter are returned as numeric values.
  Example: `SENSe:FREQuency:STOP? MAX,`
  Response: `3.5E9`

- Numeric values are output without a unit. Physical quantities are referred to the basic units or to the units set using the Unit command. The response 3.5E9 in the previous example stands for 3.5 GHz.

- Truth values (Boolean values) are returned as 0 (for OFF) and 1 (for ON).
  Example:
  Setting command: `HCOPy:DEV:COL ON`
  Query: `HCOPy:DEV:COL?`
  Response: `1`

- Text (character data) is returned in a short form.
  Example:
  Setting command: `HardCOPy:PAGE:ORIentation LANDscape`
  Query: `HCOP:PAGE:ORI?`
  Response: `LAND`

1.5 Command Sequence and Synchronization

IEEE 488.2 defines a distinction between overlapped and sequential commands:

- A sequential command is one which finishes executing before the next command starts executing. Commands that are processed quickly are usually implemented as sequential commands.

- An overlapping command is one which does not automatically finish executing before the next command starts executing. Usually, overlapping commands take longer to process and allow the program to do other tasks while being executed. If overlapping commands do have to be executed in a defined order, e.g. in order to avoid wrong measurement results, they must be serviced sequentially. This is called synchronization between the controller and the instrument.

Setting commands within one command line, even though they may be implemented as sequential commands, are not necessarily serviced in the order in which they have been received. In order to make sure that commands are actually carried out in a certain order, each command must be sent in a separate command line.

Example: Commands and queries in one message

The response to a query combined in a program message with commands that affect the queried value is not predictable. The following commands always return the specified result:

```
:FREQ:STAR 1GHZ;SPAN 100 :FREQ:STAR?
```

Result: `1000000000` (1 GHz)

Whereas the result for the following commands is not specified by SCPI:

```
:FREQ:STAR 1GHz;STAR?;SPAN 1000000
```

The result could be the value of STARt before the command was sent since the instrument might defer executing the individual commands until a program message terminator is received. The result could also be 1 GHz if the instrument executes commands as they are received.

As a general rule, send commands and queries in different program messages.

Example: **Overlapping command with *OPC**

The instrument implements SINGle as an overlapped (asynchronous) command. Assuming that SINGle takes longer to execute than *OPC, sending the following command sequence results in initiating a sweep and, after some time, setting the OPC bit in the ESR:

```
SINGle; *OPC.
```

Sending the following commands still initiates a sweep:

```
SINGle; *OPC; *CLS
```

However, since the operation is still pending when the instrument executes *CLS, forcing it into the „Operation Complete Command Idle" State (OCIS), *OPC is effectively skipped. The OPC bit is not set until the instrument executes another *OPC command.

1.5.1 Preventing Overlapping Execution

To prevent an overlapping execution of commands, one of the commands *OPC, *OPC? or *WAI can be used. All three commands cause a certain action only to be carried out after the hardware has been set. By suitable programming, the controller can be forced to wait for the corresponding action to occur.

Table 1.7: Synchronization using *OPC, *OPC? and *WAI

Command	Action	Programming the controller
*OPC	Sets the Operation Complete bit in the ESRafter all previous commands have been executed.	– Setting bit 0 in the ESE – Setting bit 5 in the SRE – Waiting for service request (SRQ)
*OPC?	Stops command processing until 1 is returned.This is only the case after the Operation Complete bit has been set in the ESR. This bit indicates that the previous setting has been completed.	Sending *OPC? directly after the commandwhose processing should be terminated beforeother commands can be executed.
*WAI	Stops further command processing until allcommands sent before *WAI have been executed.	Sending *WAI directly after the commandwhose processing should be terminated beforeother commands are executed

.

Command synchronization using *WAI or *OPC? appended to an overlapped command is a good choice if the overlapped command takes only little time to process. The two synchronization techniques simply block overlapped execution of the command.

For time consuming overlapped commands it is usually desirable to allow the controller or the instrument to do other useful work while waiting for command execution. Use one of the following methods:

***OPC with a service request**

1. Set the OPC mask bit (bit no. 0) in the ESE: *ESE 1

2. Set bit no. 5 in the SRE: *SRE 32 to enable ESB service request.

3. Send the overlapped command with *OPC

4. Wait for a service request
   The service request indicates that the overlapped command has finished.

***OPC? with a service request**

1. Set bit no. 4 in the SRE: *SRE 16 to enable MAV service request.
2. Send the overlapped command with *OPC?
3. Wait for a service request
   The service request indicates that the overlapped command has finished.

Event Status Register (ESE)

1. Set the OPC mask bit (bit no. 0) in the ESE: *ESE 1
2. Send the overlapped command without *OPC, *OPC? or *WAI
3. Poll the operation complete state periodically (by means of a timer) using the sequence: *OPC; *ESR?
   A return value (LSB) of 1 indicates that the overlapped command has finished.

***OPC? with short timeout**

1. Send the overlapped command without *OPC, *OPC? or *WAI
2. Poll the operation complete state periodically (by means of a timer) using the sequence: <short timeout>; *OPC?
3. A return value (LSB) of 1 indicates that the overlapped command has finished. In case of a timeout, the operation is ongoing.
4. Reset timeout to former value
5. Clear the error queue with `SYStem:ERRor?` to remove the „-410, Query interrupted" entries.

Using several threads in the controller application

As an alternative, provided the programming environment of the controller application supports threads, separate threads can be used for the application GUI and for controlling the instrument(s) via SCPI.

A thread waiting for a *OPC? thus will not block the GUI or the communication with other instruments.

1.6 Status Reporting System

The status reporting system stores all information on the current operating state of the instrument, and on errors which have occurred. This information is stored in the status registers and in the error queue. Both can be queried via GPIB bus or LAN interface (`STATus`... commands).

1.6.1 Structure of a SCPI Status Register

Each standard SCPI register consists of 5 parts. Each part has a width of 16 bits and has different functions. The individual bits are independent of each other, i.e. each hardware status is assigned a bit number which is valid for all five parts. Bit 15 (the most significant bit) is set to zero for all parts. Thus the contents of the register parts can be processed by the controller as positive integers.

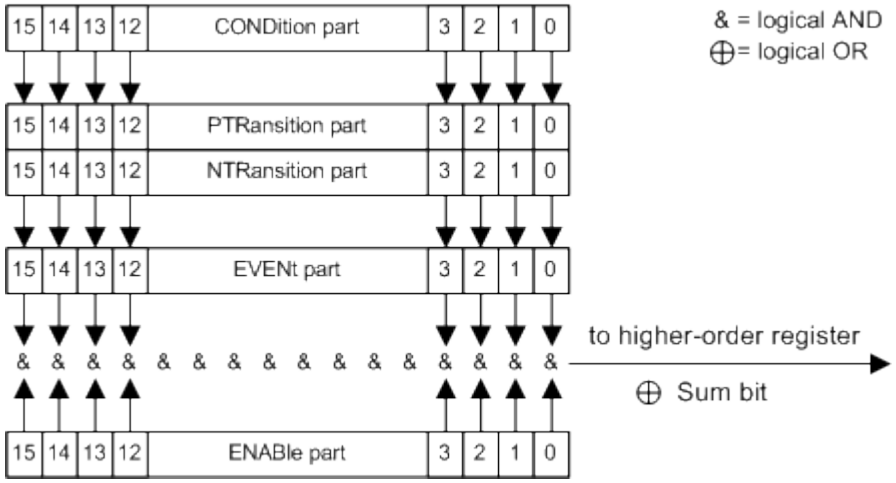


Fig. 1.4: The status-register model

Description of the five status register parts

The five parts of a SCPI register have different properties and functions:

- **CONDition**
  The `CONDition` part is written into directly by the hardware or the sum bit of the next lower register. Its contents reflect the current instrument status. This register part can only be read, but not written into or cleared. Its contents are not affected by reading.

- **PTRansition**
  The two transition register parts define which state transition of the `CONDition` part (none, 0 to 1, 1 to 0 or both) is stored in the EVENt part.
  The `Positive-TRansition` part acts as a transition filter. When a bit of the `CONDition` part is changed from 0 to 1, the associated `PTR` bit decides whether the `EVENt` bit is set to 1.
  - PTR bit =1: the `EVENt` bit is set.
  - PTR bit =0: the `EVENt` bit is not set.

  This part can be written into and read as required. Its contents are not affected by reading.

- **NTRansition**
  The `Negative-TRansition` part also acts as a transition filter. When a bit of the `CONDition` part is changed from 1 to 0, the associated `NTR` bit decides whether the `EVENt` bit is set to 1.
  - NTR bit =1: the `EVENt` bit is set.
  - NTR bit =0: the `EVENt` bit is not set.

  This part can be written into and read as required. Its contents are not affected by reading.

- **EVENt**
  The `EVENt` part indicates whether an event has occurred since the last reading, it is the „memory" of the condition part. It only indicates events passed on by the transition filters. It is permanently updated by the instrument. This part can only be read by the user. Reading the register clears it. This part is often equated with the entire register.

- **ENABle**
  The `ENABle` part determines whether the associated `EVENt` bit contributes to the sum bit (see below). Each bit of the EVENt part is „ANDed" with the associated ENABle bit (symbol ‚&'). The results of all logical operations of this part are passed on to the sum bit via an „OR" function (symbol ‚+').
  `ENABle` bit = 0: the associated `EVENt` bit does not contribute to the sum bit `ENABle` bit = 1: if the associated `EVENt` bit is „1", the sum bit is set to „1" as well. This part can be written into and read by the user as required. Its contents are not affected by reading.

Sum bit

The sum bit is obtained from the EVENt and ENABle part for each register. The result is then entered into a bit of the CONDition part of the higher-order register.

The instrument automatically generates the sum bit for each register. Thus an event can lead to a service request throughout all levels of the hierarchy.

1.6.2 Hierarchy of status registers

As shown in the following figure, the status information is of hierarchical structure.

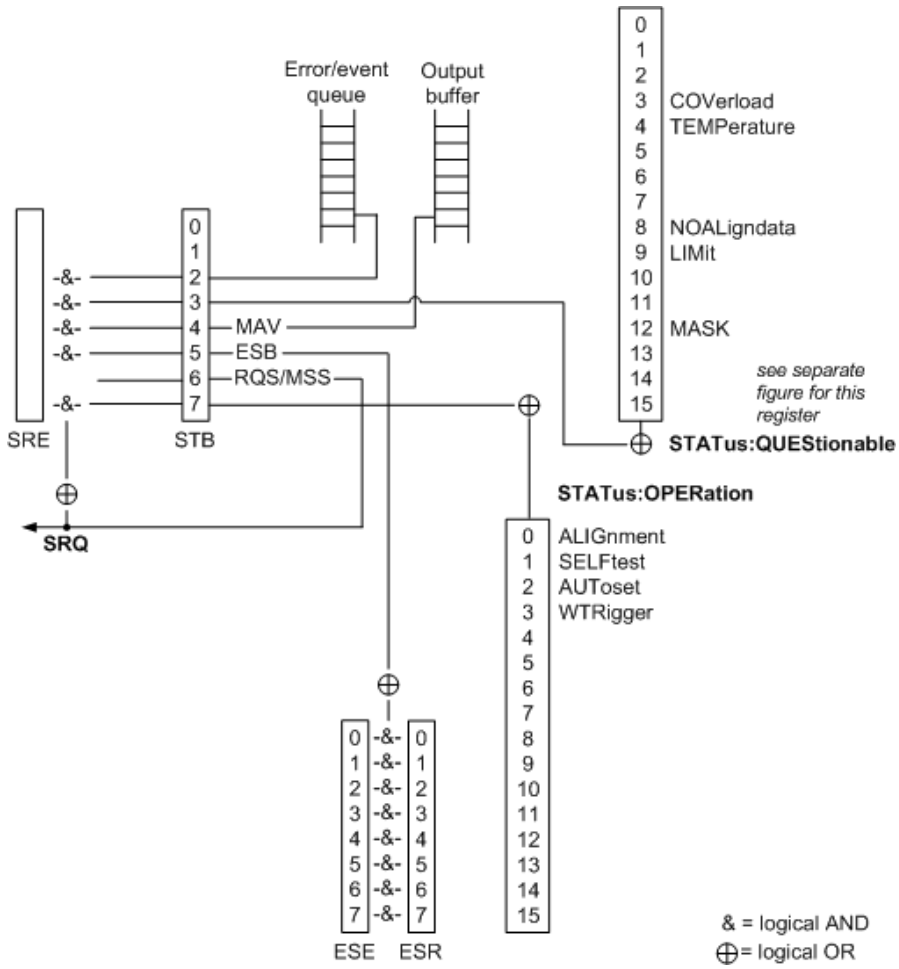


Fig. 1.5: Overview of the status registers hierarchy

- **STB, SRE**
  The STatus Byte (STB) register and its associated mask register Service Request Enable (SRE) form the highest level of the status reporting system. The STB provides a rough overview of the instrument status, collecting the information of the lower-level registers.

- **ESR, SCPI registers**
  The STB receives its information from the following registers:
  - The Event Status Register (ESR) with the associated mask register standard Event Status Enable (ESE).
  - The STATus:OPERation and STATus:QUEStionable registers which are defined by SCPI and contain detailed information on the instrument.

- **Output buffer**
  The output buffer contains the messages the instrument returns to the controller. It is not part of the status reporting system but determines the value of the MAV bit in the STB and thus is represented in the overview.

All status registers have the same internal structure.

SRE, ESE
The service request enable register SRE can be used as ENABle part of the STB if the STB is structured according to SCPI. By analogy, the ESE can be used as the ENABle part of the ESR.

1.6.3 Contents of the Status Registers

In the following sections, the contents of the status registers are described in more detail.

Status Byte (STB) and Service Request Enable Register (SRE)

The STatus Byte (STB) is already defined in IEEE 488.2. It provides a rough overview of the instrument status by collecting the pieces of information of the lower registers. A special feature is that bit 6 acts as the sum bit of the remaining bits of the status byte.

The STB can thus be compared with the CONDition part of an SCPI register and assumes the highest level within the SCPI hierarchy.

The STB is read using the command *STB or a serial poll.

The STatus Byte (STB) is linked to the Service Request Enable (SRE) register. Each bit of the STB is assigned a bit in the SRE. Bit 6 of the SRE is ignored. If a bit is set in the SRE and the associated bit in the STB changes from 0 to 1, a service request (SRQ) is generated. The SRE can be set using the command *SRE and read using the command *SRE?.

Table 1.8: Meaning of the bits used in the status byte

Bit No.	Meaning
0...1	Not used
2	Error Queue not empty The bit is set when an entry is made in the error queue. If this bit is enabled by the SRE, each entry of the error queue generates a service request. Thus an error can be recognized and specified in greater detail by polling the error queue. The poll provides an informative error message. This procedure is to be recommended since it considerably reduces the problems involved with remote control.
3	QUEStionable status sum bit The bit is set if an EVENt bit is set in the QUEStionable status register and the associated ENABle bit is set to 1. A set bit indicates a questionable instrument status, which can be specified in greater detail by polling the QUEStionable status register.
4	MAV bit (message available) The bit is set if a message is available in the output buffer which can be read. This bit can be used to enable data to be automatically read from the instrument to the controller.
5	ESB bit Sum bit of the event status register. It is set if one of the bits in the event status register is set and enabled in the event status enable register. Setting of this bit indicates a serious error which can be specified in greater detail by polling the event status register.
6	MSS bit (master status summary bit) The bit is set if the instrument triggers a service request. This is the case if one of the other bits of this registers is set together with its mask bit in the service request enable register SRE.
7	OPERation status register sum bit The bit is set if an EVENt bit is set in the OPERation status register and the associated ENABle bit is set to 1. A set bit indicates that the instrument is just performing an action. The type of action can be determined by polling the OPERation status register.

Event Status Register (ESR) and Event Status Enable Register (ESE)

The ESR is defined in IEEE 488.2. It can be compared with the EVENt part of a SCPI register. The event status register can be read out using command *ESR?.

The ESE corresponds to the ENABle part of a SCPI register. If a bit is set in the ESE and the associated bit in the ESR changes from 0 to 1, the ESB bit in the STB is set. The ESE register can be set using the command *ESE and read using the command *ESE?.

Table 1.9: Meaning of the bits used in the event status register

Bit No.	Meaning
0	Operation Complete This bit is set on receipt of the command *OPC exactly when all previous commands have been executed.
1	Not used
2	Query Error This bit is set if either the controller wants to read data from the instrument without having sent a query, or if it does not fetch requested data and sends new instructions to the instrument instead. The cause is often a query which is faulty and hence cannot be executed.
3	Device-dependent Error This bit is set if a device-dependent error occurs. An error message with a number between -300 and -399 or a positive error number, which denotes the error in greater detail, is entered into the error queue.
4	Execution Error This bit is set if a received command is syntactically correct but cannot be performed for other reasons. An error message with a number between -200 and -300, which denotes the error in greater detail, is entered into the error queue.
5	Command Error This bit is set if a command is received, which is undefined or syntactically incorrect. An error message with a number between -100 and -200, which denotes the error in greater detail, is entered into the error queue.
6	User Request This bit is set when the instrument is switched over to manual control.
7	Power On (supply voltage on) This bit is set on switching on the instrument.

STATus:OPERation Register

In the CONDition part, this register contains information on which actions the instrument is being executing. In the EVENt part, it contains information on which actions the instrument has executed since the last reading. It can be read using the commands STATus:OPERation:CONDition? or STATus:OPERation[:EVENt]?.

See also figure 1.5: „Overview of the status registers hierarchy“

The remote commands for the STATus:OPERation register are described in chapter 2.14.1: „STATus:OPERation Register“ on pageSeite 168.

Table 1.10: Bits in the STATus:OPERation register

Bit No.	Meaning
0	ALIGnment This bit is set as long as the instrument is performing a self alignment.
1	SELFtest This bit is set while the selftest is running.
2	AUToset This bit is set while the instrument is performing an auto setup.
3	WTRIgger This bit is set while the instrument is waiting for the trigger.
4 to 14	Not used
15	This bit is always 0.

STATus:QUEStionable Register

This register contains information about indefinite states which may occur if the unit is operated without meeting the specifications. It can be read using the commands „STATus:QUEStionable:CONDition?" on pageSeite 170 and „STATus:QUEStionable[:EVENt]?" on pageSeite 171.

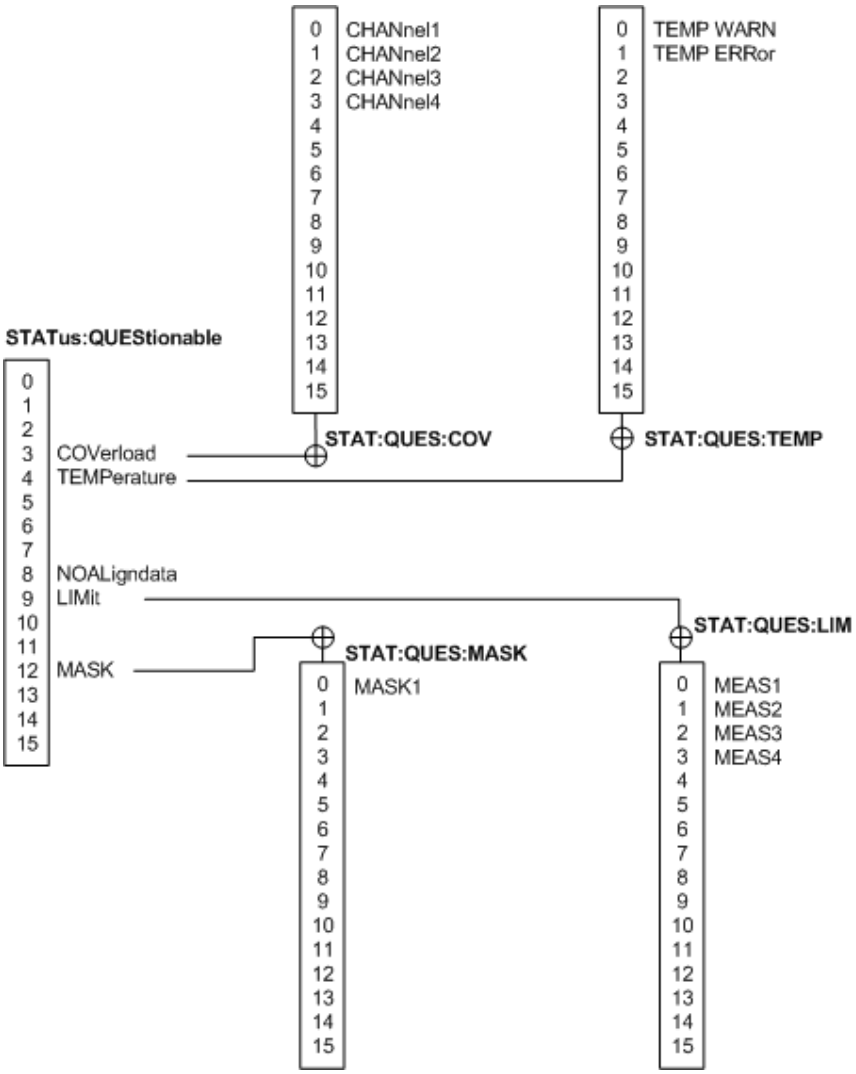


Fig. 1.6: Overview of the STATus:QUEStionable register

Table 1.11: Bits in the STATus:QUEStionable register

Bit No.	Meaning
0 to 2	not used
3	COVerload This bit is set if a questionable channel overload occurs (see: STATus:QUEStionable:COVerload register, on the page below).
4	TEMPerature This bit is set if a questionable temperature occurs (see „STATus:QUEStionable:TEMPerature register“, on the page below).
5 to 7	Not used
8	NOALigndata This bit is set if no alignment data is available - the instrument is uncalibrated.
9	LIMit This bit is set if a limit value is violated (see: STATus:QUEStionable:LIMit register, next page).
10 to 11	Not used
12	MASK This bit is set if a mask value is violated (see: STATus:QUEStionable:MASK register, next page).
13 to 14	Not used
15	This bit is always 0.

STATus:QUEStionable:COVerload register

This register contains all information about overload of the channels. The bit is set if the assigned channel is overloaded.

Table 1.12: Bits in the STATus:QUEStionable:COVerload register

Bit No.	Meaning
0	CHANnel1
1	CHANnel2
2	CHANnel3
3	CHANnel4

STATus:QUEStionable:TEMPerature register

This register contains information about the instrument's temperature.

Table 1.13: Bits in the STATus:QUEStionable:TEMPerature register

Bit No.	Meaning
0	TEMP WARN This bit is set if a temperature warning on channel 1, 2, 3 or 4 occured.
1	TEMP ERRor This bit is set if a temperature error on channel 1, 2, 3 or 4 occured.

STATus:QUEStionable:LIMit register

This register contains information about the observance of the limits of measurements.This bit is set if the limits of the main or additional measurement of the assigned measurement are violated.

Table 1.14: Bits in the STATus:QUEStionable:LIMit register

Bit No.	Meaning
0	MEAS1
1	MEAS2
2	MEAS3
3	MEAS4

STATus:QUEStionable:MASK register

This register contains information about the violation of masks. This bit is set if the assigned mask is violated.

Table 1.15: Bits in the STATus:QUEStionable:MASK register

Bit No.	Meaning
0	MASK1

1.6.4 Application of the Status Reporting System

The purpose of the status reporting system is to monitor the status of one or several devices in a measuring system. To do this and react appropriately, the controller must receive and evaluate the information of all devices. The following standard methods are used:

- **Service request** (SRQ) initiated by the instrument
- **Serial poll** of all devices in the bus system, initiated by the controller in order to find out who sent a SRQ and why
- **Parallel poll** of all devices
- Query of a **specific instrument status** by means of commands
- Query of the **error queue**

Service Request

Under certain circumstances, the instrument can send a service request (SRQ) to the controller. Usually this service request initiates an interrupt at the controller, to which the control program can react appropriately. As evident from figure 1.5, an SRQ is always initiated if one or several of bits 2, 3, 4, 5 or 7 of the status byte are set and enabled in the SRE. Each of these bits combines the information of a further register, the error queue or the output buffer. The ENABle parts of the status registers can be set such that arbitrary bits in an arbitrary status register initiate an SRQ. In order to make use of the possibilities of the service request effectively, all bits should be set to „1" in enable registers SRE and ESE.

The SRQ is the only possibility for the instrument to become active on its own. Each controller program should cause the instrument to initiate a service request if errors occur. The program should react appropriately to the service request.

Serial Poll

In a serial poll, just as with command *STB, the status byte of an instrument is queried. However, the query is realized via interface messages and is thus clearly faster. The serial poll method is defined in IEEE 488.1 and used to be the only standard possibility for different instruments to poll the status byte. The method also works for instruments which do not adhere to SCPI or IEEE 488.2.

The serial poll is mainly used to obtain a fast overview of the state of several instruments connected to the controller.

Query of an instrument status

Each part of any status register can be read using queries. There are two types of commands:

– The common commands `*ESR?, *IDN?, *IST?, *STB?` query the higher-level registers.

– The commands of the STATus system query the SCPI registers (`STATus:QUEStionable...`)

The returned value is always a decimal number that represents the bit pattern of thequeried register. This number is evaluated by the controller program.

Queries are usually used after an SRQ in order to obtain more detailed information on the cause of the SRQ.

Decimal representation of a bit pattern

The STB and ESR registers contain 8 bits, the SCPI registers 16 bits. The contents of a status register are specified and transferred as a single decimal number. To make this possible, each bit is assigned a weighted value. The decimal number is calculated as the sum of the weighted values of all bits in the register that are set to 1.

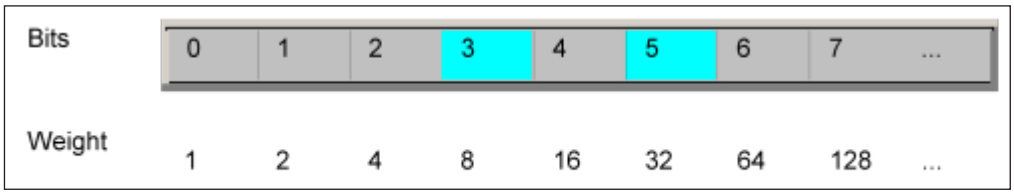


Fig. 1.7: Decimal representation of a bit pattern

Example:
The decimal value 40 = 32 + 8 indicates that bits no. 3 and 5 in the status register (e.g. the `QUEStionable` status summary bit and the `ESB` bit in the `STatus` Byte) are set.

Error Queue

Each error state in the instrument leads to an entry in the error queue. The entries of the error queue are detailed plain text error messages that can be looked up in the Error Log or queried via remote control using `SYSTem:ERRor[:NEXT]?` or `SYSTem:ERRor:ALL?`. Each call of `SYSTem:ERRor[:NEXT]?` provides one entry from the error queue. If no error messages are stored there any more, the instrument responds with 0, „No error".

The error queue should be queried after every SRQ in the controller program as the entries describe the cause of an error more precisely than the status registers. Especially in the test phase of a controller program the error queue should be queried regularly since faulty commands from the controller to the instrument are recorded there as well.

1.6.5 Reset Values of the Status Reporting System

The following table contains the different commands and events causing the status reporting system to be reset. None of the commands, except *RST and SYSTem:PRESet, influence the functional instrument settings. In particular, DCL does not change the instrument settings.

Table 1.16: Resest of the status reporting system

Event	Switching on supply-voltage Power-On-Status-Clear		DCL, SDC (DeviceClear,SelectedDevice-Clear)	*RST orSYSTem:PRESet	STATus:PRESet	*CLS
Effect	0	1				
Clear STB, ESR	-	yes	-	-	-	yes
Clear SRE, ESE	-	yes	-	-	-	-
Clear EVENt parts of the registers	-	yes	-	-	-	yes
Clear ENABle parts of all OPERation and QUEStionable registers; Fill ENABle parts of all other registers with „1“.	-	yes	-	-	yes	-
Fill PTRansition parts with „1“; Clear NTRansition parts	-	yes	-	-	yes	-
Clear error queue	yes	yes	-	-	-	yes
Clear output buffer	yes	yes	yes	1)	1)	1)
Clear command processing and input buffer	yes	yes	yes	-	-	-
1) The first command in a command line that immediately follows a <PROGRAM MESSAGE TERMINATOR> clears the output buffer.						

1.7 General Programming Recommendations

Initial instrument status before changing settings

Manual operation is designed for maximum possible operating convenience. In contrast, the priority of remote control is the „predictability“ of the instrument status. Thus, when a command attempts to define incompatible settings, the command is ignored and the instrument status remains unchanged, i.e. other settings are not automatically adapted. Therefore, control programs should always define an initial instrument status (e.g. using the *RST command) and then implement the required settings.

Command sequence

As a general rule, send commands and queries in different program messages. Otherwise, the result of the query may vary depending on which operation is performed first (see also Preventing Overlapping Execution on page 17).

Reacting to malfunctions

The service request is the only possibility for the instrument to become active on its own. Each controller program should instruct the instrument to initiate a service request in case of malfunction. The program should react appropriately to the service request.

Error queues

The error queue should be queried after every service request in the controller program as the entries describe the cause of an error more precisely than the status registers. Especially in the test phase of a controller program the error queue should be queried regularly since faulty commands from the controller to the instrument are recorded there as well.

2 Command Reference

This chapter provides the description of all remote commands available for oscilloscopes HMO series. The commands are sorted according to the menu structure of the instrument. A list of commands in alphabetical order ist given in the „List of Commands" at the end of this documentation.

2.1 Common Commands

Common commands are described in the IEEE 488.2 (IEC 625-2) standard. These commands have the same effect and are employed in the same way on different devices.The headers of these commands consist of „*" followed by three letters. Many common commands are related to the Status Reporting System.

Available common commands:.

***CAL?**
Calibration Query

Initiates a calibration of the instrument and subsequently queries the calibration status. Responses > 0 indicate errors.

***CLS**
CLear Status

Sets the status byte (STB), the standard event register (ESR) and the EVENt part of the QUEStionable and the OPERation registers to zero. The command does not alter the mask and transition parts of the registers. It clears the output buffer.

Usage: Setting only

***ESE** <Value>
Event Status Enable

Sets the event status enable register to the specified value. The query returns the contents of the event status enable register in decimal form.

Parameters:
<Value> **Range:** 0 to 255

***ESR?**
Event Status Read
Returns the contents of the event status register in decimal form and subsequently sets the register to zero.

Return values:
<Contents> **Range:** 0 to 255

Usage: Query only

***IDN?**
IDeNtification:returns the instrument identification.

Return values:
<ID> HAMEG,<device type>,<serial number>,<firmwareversion>

Example: HAMEG,HMO2024,12345,0.10.1.23

Usage: Query only

***OPC**
OPeration Complete

Sets bit 0 in the event status register when all preceding commands have been executed. This bit can be used to initiate a service request. The query form writes a „1" into the output buffer as soon as all preceding commands have been executed. This is used for command synchronization.

***OPT?**
OPTion identification query

Queries the options included in the instrument. For a list of all available options and their description refer to the CD-ROM.

Return values:
<Options> The query returns a list of options. A zero is returned for options that are not installed.

Usage: Query only

***PSC** <Action>

Power on Status Clear

Determines whether the contents of the ENABle registers are preserved or reset when the instrument is switched on. Thus a service request can be triggered when the instrument is switched on, if the status registers ESE and SRE are suitably configured. The query reads out the contents of the „power-on-status-clear" flag.

Parameters:
<Action> 0 | 1
0
The contents of the status registers are preserved.
1
Resets the status registers.

***RST**
ReSeT

Sets the instrument to a defined default status. The default settings are indicated in the description of commands.

Usage: Setting only

***SRE** <Contents>
Service Request Enable

Sets the service request enable register to the indicated value. This command determines under which conditions a service request is triggered.

Parameters:

<Contents>	Contents of the service request enable register in decimal form. Bit 6 (MSS mask bit) is always 0. Range: 0 to 255

***STB?**
STatus Byte query

Reads the contents of the status byte in decimal form.

Usage: Query only

***TRG**
TRiGger

Triggers all actions waiting for a trigger event. In particular, *TRG generates a manual trigger signal (Manual Trigger). This common command complements the commands of the `TRIGger` subsystem.

Usage: Event

***TST?**
self TeST query

Triggers selftests of the instrument and returns an error code in decimal form (see Service Manual supplied with the instrument). „0" indicates no errors occured.

Usage: Query only

***WAI**
WAIt to continue

Prevents servicing of the subsequent commands until all preceding commands have been executed and all signals have settled (see also command synchronization and `*OPC`).

Usage: Event

2.2 Acquisition and Setup

2.2.1 Starting and Stopping Acquisition

RUN
Starts the continuous acquisition.

Usage: Event
Asynchronous command

RUNContinous
Same as RUN.

Usage: Event
Asynchronous command

SINGle
Starts a defined number of acquisition cycles.

Usage: Event
Asynchronous command

RUNSingle
Same as SINGle.

Usage: Event
Asynchronous command

STOP
Stops the running acquistion.

Usage: Event
Asynchronous command

2.2.2 Time Base

TIMebase:SCALe <TimeScale>
Sets the horizontal scale for all channel and math waveforms.

Parameters:

<TimeScale>	Range:	1e-9 to 50
	Increment:	1e-9
	*RST:	100e-6
	Default unit:	s/div

TIMebase:RATime?
Queries the real acquisition time used in the hardware. If FFT analysis is performed, the value can differ from the adjusted acquisition time (`TIMebase:ACQTime`).

Return values:

<HWAcqTime>	Increment:	1e-12
	Default unit:	s

Usage: Query only

TIMebase:ACQTime <AcquisitionTime>
Defines the time of one acquisition, that is the time across the 12 divisions of the diagram:**Timebase Scale*12.**

Parameters:

<AcquisitionTime>	Range:	12ns to 600s
	Increment:	1e-12
	Default unit:	s

TIMebase:RANGe <AcquisitionTime>
Defines the time of one acquisition, that is the time across the 12 divisions of the diagram: **Timebase Scale*12.**

Parameters:

<AcquisitionTime>	Range:	12ns to 600s
	Default unit:	s

TIMebase:DIVisions?
Queries the number of horizontal divisions on the screen.

Return values:

<HorizDivCount>	Range:	12
	*RST:	12

Usage: Query only

TIMebase:POSition <Offset>
Defines the trigger position (trigger offset) - the time interval between trigger point and reference point to analize the signal some time before or after the trigger event.

See also: `TIMebase:REFerence`

Parameters:

<Offset>	Range:	-500 to 500
	Increment:	0.01 (depending on timebase)
	*RST:	0
	Default unit:	s

TIMebase:REFerence <ReferencePoint>
Sets the reference point of the time scale (Time Reference) in % of the display. The reference point defines which part of the waveform is shown. If the trigger position is zero, the trigger point matches the reference point.

See also: `TIMebase:POSition`

Parameters:

<ReferencePoint>	Range:	10 to 90
	Increment:	10
	*RST:	50
	Default unit:	%

2.2.3 Acquisition

AUToscale
Performs an autoset process, analyzes the enabled channel signals, and obtains appropriate horizontal, vertical, and trigger settings to display stable waveforms.

Usage: Event
Asynchronous command

ACQuire:MODE <AcquisitionMode>
Selects the method of adding waveform points to the samples of the ADC in order to fill the record length.

Parameters:
<AcquisitionMode> RTIMe | ETIMe

RTIMe
Real Time Mode:At slow time base settings the sampled points of the input signal are used to build the waveform, no waveform points are added. With fast time base settings, the sample rate is higher than the ADC sample rate. Waveform samples are added to the ADC samples with sin(x)/x interpolation.

ETIMe
Equivalent time:The waveform points are taken from several acquisitions of a repetive signal at a different time in relation to the trigger point.
*RST: RTIME

ACQuire:INTerpolate <Interpolation>
Defines the interpolation mode.

Parameters:
<Interpolation> LINear | SINX | SMHD

LINear: Linear interpolation between two adjacent sample points.
SINX: Interpolation by means of a sin(x)/x curve.
SMHD: Sample & Hold causes a histogram-like interpolation.

*RST: SINX

ACQuire:AVERage:COUNt <AverageCount>
Defines the number of waveforms used to calculate the average waveform. The higher the number, the better the noise is reduced.

Parameters:
<AverageCount> Only numbers from the 2nd progression are permitted.

ACQuire:WRATe <WaveformRate>
Defines the mode to set the sample rate (samples per second saved in the memory) and the waveform acquisition rate (waveforms per second).

Parameters:
<WaveformRate> AUTO | MWAVeform | MSAMples

AUTO
To display the best waveform, the instrument selects the optimum combination of waveform acquisition rate and sample rate using the full memory depth.

MWAVeform
Maximum waveform rate:The instrument combines sample rate and memory depth to acquire at maximum waveform acquisition rate. In connection with persistence, the mode can display rare signal anomalies.

MSAMples
Maximum sample rate:The instrument acquires the signal at maximum sample rate and uses the full memory depth. The result is a waveform with maximum number of waveform samples, high degree of accuracy, and low risk of aliasing.

*RST: AUTO

CHANnel<**m**>**:TYPE** <DecimationMode>
Selects the method to reduce the data stream of the ADC to a stream of waveform points with lower sample rate.

Suffix:
<m> The command affects all channels regardless of the indicated channel number.
The suffix can be omitted.

Parameters:
<DecimationMode> SAMPle | PDETect | HRESolution

SAMPle
Input data is acquired with a sample rate which is aligned to the time base (horizontal scale) and the record length.

PDETect
Peak Detect:the minimum and the maximum of n samples in a sample interval are recorded as waveform points.

HRESolution
High resolution:The average of n sample points is recorded as waveform point.

*RST: SAMPle

CHANnel<**m**>**:ARIThmetics** <TrArithmetic>
Selects the method to build the resulting waveform from several consecutive acquisitions of the signal.

Suffix:
<m> The command affects all channels regardless of the indicated channel number.
The suffix can be omitted.

Parameters:
<TrArithmetic> OFF | ENVelope | AVERage | FILTer

OFF
The data of the current acquisition is recorded according to the decimation settings.

ENVelope
Detects the minimum and maximum values in an sample interval over a number of acquisitions.

AVERage
Calculates the average from the data of the current acquisition and a number of acquisitions before. The number of used acquisitions is set with `ACQuire:AVERage:COUNt.`

FILTer
Sets a low-pass filter with 3 db attenuation at a configurable limit frequency set with `ACQuire:FILTer:FREQency.`
The filter removes higher frequencies from the channel signals.

*RST: OFF

TIMebase:ROLL:ENABle <Roll>
Enables the roll mode.

Parameters:
<Roll> ON | OFF

*RST: OFF

ACQuire:FILTer:FREQency <FilterFrequency>
Sets the limit frequency if `CHANnel<m>:ARIThmetics` is set to „FILTer".

Parameters:

<FilterFrequency>	Limit frequency with 3 dB attenuation Default unit: Hz

ACQuire:POINts:ARATe?

Retrieves the sample rate of the ADC, that is the number of points that are sampled by the ADC in one second.

Return values:

<AcquisitionRate>	ADC sample rate Range: 2.5E3 to 4E9 (depending on HMO model) Increment: 1E3 *RST: 4E9 (depending on HMO model) Default unit: Hz

Usage: Query only

ACQuire:SRATe?
Returns the sample rate, that is the number of recorded waveform samples per second.

Return values:

<SampleRate>	Range: 2 to 1E11 Increment: depends on time base, waveform rate, number of active channels *RST: 1E7 Default unit: Sa/s

Usage: Query only

ACQuire:STATe <Aquisition State>
Sets the aquisition state of the instrument.

Parameters:

<Aquisition State> RUN | STOP | COMPlete | BREak

RUN
The aquisition is running.

STOP
The acquisition will stop if finished.

COMPlete
The current aquisition is finished and completed.

BREak
Set: will break/interrupt current acquisition
Read: acquisition is finished but interrupted

*RST: RUN

ACQuire:TYPE <Aquisition Type>
Sets the type of the aquisition mode.

Parameters:
<Aquisition Type> REFresh | ROLL | AVERage | ENVelope | FILTer

REFresh
The aquisitions are displayed as they are done.

ROLL
The aquisitions are done in roll mode.

AVERage
The aquisitions are averaged.

FILTer
Sets a low-pass filter with 3 db attenuation at a configurable limit frequency

*RST: OFF

ACQuire:REALtime <Random Sampling>
Enables the random sampling.

Parameters:
<Random Sampling> AUTO | OFF

*RST: OFF

ACQuire:PEAKdetect <PeakDetect>
Enables the peak detection.

Parameters:
<PeakDetect> AUTO | OFF

*RST: OFF

ACQuire:HRESolution <HighRes>
Enables the high resolution mode.

Parameters:
<HighRes> AUTO | OFF

*RST: OFF

2.2.4 Vertical

CHANnel<m>:STATe <State>
Switches the channel signal on or off.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<State> ON | OFF

CHANnel<m>:COUPling <Coupling>
Selects the connection of the indicated channel signal.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Coupling> DC | DCLimit | AC | ACLimit

DC: Direct connection with 50 Ω termination.
DCLimit: Direct connection with 1 MΩ termination.
AC: Connection through DC capacitor that removes the DC offset voltage from the input signal.
ACLimit: Connection through DC capacitor with 1 MΩ termination. The capacitor removes the DC offset voltage from the input signal.

*RST: DCLimit

CHANnel<m>:SCALe <Scale>
Sets the vertical scale for the indicated channel.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Scale> Scale value, given in Volts per division.
Range: 1e-3 to 10
Increment: 0.01
*RST: 5e-3
Default unit: V/div

CHANnel<**m**>**:RANGe** <Range>
Sets the voltage range across the 8 vertical divisions of the diagram. Use the command alternativly instead of `CHANnel<m>:SCALe` (see command above).

Suffix:
<m> Selects the input channel. The maximum channel number is instrument-dependent.

Parameters:
<Range> Voltage range value
Range: 8e-3 to 80
*RST: 40e-3
Default unit: V

CHANnel<**m**>**:POSition** <Position>
Sets the vertical position of the indicated channel and its horizontal axis in the window.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Position> Position value, given in divisions.
Range: -5 to 5
*RST: 0
Default unit: div

CHANnel<**m**>**:OFFSet** <Offset>
The offset voltage is subtracted to correct an offset-affected signal.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Offset> Offset value

*RST: 0

Default unit: V

CHANnel<**m**>**:BANDwidth** <BandwidthLimit>
Selects the bandwidth limit for the indicated channel.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<BandwidthLimit> FULL | B20

FULL
Use full bandwidth.

B20
Bandwidthlimit 20 MHz.

*RST: FULL

CHANnel<**m**>**:POLarity** <Polarity>
Turns the inversion of the signal amplitude on or off. To invert means to reflect the voltage values of all signal components against the ground level. Inversion affects only the display of the signal but not the trigger.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Polarity> NORMal | INVerted

CHANnel<**m**>**:OVERload** <Overload>
Retrieves the overload status of the specified channel from the status bit. When the overload problem is solved, the command resets the status bit.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Overload> ON | OFF
Use OFF to reset the overload status bit.

*RST: OFF

Example:
`CHANnel2:OVERload?`
Queries the overload status of channel 2.
`CHANnel2:OVERload OFF`
Resets the overload status bit.

CHANnel<**m**>**:SKEW** <Skew>
Skew or deskew compensates delay differences between channels caused by the different length of cables, probes, and other sources. Correct deskew values are important for accurate triggering.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Skew> Deskew value

Default unit: s

CHANnel<**m**>**:THReshold** <Threshold>
Threshold value for digitization of analog signals. If the signal value is higher than the threshold, the signal state is high (1 or true for the boolean logic). Otherwise, the signal state is considered low (0 or false) if the signal value is below the threshold.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Threshold> Default values are:
TTL: 1,4 V
ECL: -1,3 V
CMOS: 2,5 V

Default unit: V

CHANnel<**m**>**:LABel** <Label>
Set the label for the input channel.

Suffix:
<m> 1...4
Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Label> String value
String with max. 8 characters, only ASCII characters can be used.

Example: "V-Input"
"VoltagA"

CHANnel<**m**>**:LABel:STATe** <State>
Switches the label of the channel on or off

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<State> ON | OFF

2.2.5 Logic Channel

LOGic<I>:POSition <Position>
Set the position of a logic channel.

Suffix:
<I> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Position> Position value, given in divisions.

Default unit: div

LOGic<I>:SIZE <Size>
Set the size of the display of teh logic channel.

Suffix:
<I> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Size> SMALl | MEDium | LARGe

LOGic<I>:STATe <state>
Switches the channel signal on or off.

Suffix:
<I> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<State> ON | OFF

LOGic<I>:LABel <Label>
Set the label for the logic channel.

Suffix:
<I> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Label> string value
"xxxxxxxx" (maximum 8 characters)

LOGic<l>:LABel:State <Label>
Switches the label of the lagic channel on or off.

Suffix:
<l> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Label> ON | OFF

POD<p>:THReshold <Threshold Mode>
Threshold Mode for Logic Pod. If the signal value is higher than the threshold, the signal state is high (1 or true for the boolean logic). Otherwise, the signal state is considered low (0 or false) if the signal value is below the threshold.

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524..

Parameters:
<Thresholds Mode> TTL | ECL | CMOS | USER1 | USER2

TTL
TTL level, set to 1.4V

ECL
ECL level, set to -1.3V

CMOS
CMOS level, set to 2.5V

USER1
USER1 level, can be set with the POD<p>:THReshold:UDLevel<u> command.

USER2
USER2 level, can be set with the POD<p>:THReshold:UDLevel<u> command.

POD<p>:THReshold:UDLevel<u> <Threshold Level>
Set the User high level threshold for the pod.

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

<u> Select the User setting. (USER1, USER2)

Parameters:
<Thresholds Level> Range: -2 to 8
Increment: 1e-2
Default unit: V

POD<p>:STATe <State>
Switches the Pod on or off.

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Parameters:
<State> ON | OFF

POD<p>:DATA? <Data>
Returns the data of the POD. The data can be used in MATLAB, for example.

To set the export format, use „FORMat[:DATA]" on page 50.

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Return values:
<Data> List of values according to the format settings.

Usage: Query only

POD<p>:DATA:HEADer?
Returns the header of data of the POD.

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Table 2.1: Header data

Position	Meaning	Example
1	XStart in s	-9.477E-008 = - 94,77 ns
2	XStop in s	9.477E-008 = 94,77 ns
3	Number of Samples	1200
4	Number of values per sample interval. For PeakDetect waveforms the value is 2.	2

Usage: Query only

POD<p>:DATA:POINts <Points>
Returns the number of available data.

As a setting, the command selects a range of samples. As a query, it returns the number of samples for the selected range.

To get correct results with settings MAX and DMAX, the acquisition must not run when the command is used. To acquire consistent and valid data, use the SINGle command before. Do not cancel a running acquisition with STOP because the waveform data might be incomplete or incorrect. Please use the command `ACQuire:STATe STOP` to stop the acquisition.

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Return values:
<Points> Number of data points in the selected range.

Default unit: Samples

Setting Parameters:

<Points> DEFault | MAXimum | DMAXimum

Sets the range for the query.

DEFault
Waveform samples displayed on the screen.

MAXimum
Range is the complete memory (only available in STOP mode).

DMAXimum
Display maximum:number of samples in the current waveform record (only available in STOP mode).

POD<p>:DATA:XINCrement?
Return the time difference between two adjacent samples of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:

<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Return values:

<Xincrement> Time in s

Usage: Query only

POD<p>:DATA:XORigin?
Return the time of the first sample of the indicated waveform (depending on the trigger event t=0).
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:

<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Return values:

<Xorigin> Time in s

Usage: Query only

POD<p>:DATA:YINCrement?
Returns the value per bit of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:

<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Usage: Query only

POD<p>:DATA:YORigin
Return the value for binary value 0 of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

POD<p>:DATA:YRESolution?
Return the vertical bit resolution of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<p> Select the Pod.
The possible number of Pods are **1** for HMO72x...202x and **2** for HMO352x/HMO2524.

Return values:
<Yresolution> For default waveforms, the resolution is 8 bit.

Usage: Query only

2.2.6 Waveform Data

CHANnel<m>:DATA?
Returns the data of the channel waveform points. The waveforms data can be used in MATLAB, for example.

For envelope waveforms, use the `CHANnel<m>:DATA:ENVelope` command.

To set the export format, use „`FORMat[:DATA]`" on page 50.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Data> List of values according to the format settings.

Example: CHAN1:DATA?
-0.125000,-0.123016,-0.123016,-0.123016, -0.123016,-0.123016,...

Usage: Query only

CHANnel<m>:DATA:HEADer?
Returns information on the channel waveform.
For envelope waveforms, use the `CHANnel<m>:DATA:ENVelope:HEADer` command.

Table 2.2: Header data

Position	Meaning	Example
1	XStart in s	-9.477E-008 = - 94,77 ns
2	XStop in s	9.477E-008 = 94,77 ns
3	Record length of the waveform in Samples	200000
4	Number of values per sample interval. For PeakDetect waveforms the value is 2.	2

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<DataHeader> Comma-separated value list

Example:
```
CHAN:DATA:HEAD?
-9.477E-008,9.477E-008,200000,1
```

Usage: Query only

CHANnel<m>:DATA:ENVelope?
Returns the data of the envelope waveform.

Use this command only for envelope waveforms. for all other channel waveforms use CHANnel<m>:DATA.

To set the export format, use „FORMat[:DATA]" on page 50.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Data> List of values according to the format settings.

Usage: Query only

CHANnel<m>:DATA:ENVelope:HEADer?
Returns information on the envelope waveform.

Use this command only for envelope waveforms. for all other channel waveforms use `CHANnel<m>:DATA:HEADer`.

Table 2.3: Header data

Position	Meaning	Example
1	XStart in s	-9.477E-008 = - 94,77 ns
2	XStop in s	9.477E-008 = 94,77 ns
3	Number of Samples	1200
4	Number of values per sample interval. For envelope waveforms the value is 2.	2

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<DataHeader> Comma-separated value list

Example: -9.477E-008,9.477E-008,1200,2

Usage: Query only

CHANnel<m>:DATA:XINCrement?
CHANnel<m>:ENVelope:DATA:XINCrement?
Return the time difference between two adjacent samples of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Xincrement> Time in s

Usage: Query only

CHANnel<m>:DATA:XORigin?
CHANnel<m>:DATA:ENVelope:XORigin?
Return the time of the first sample of the indicated waveform (depending on the trigger event t=0).
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Xorigin> Time in s

Usage: Query only

CHANnel<m>:DATA:YINCrement?
CHANnel<m>:DATA:ENVelope:YINCrement?
Return the value per bit of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Yincrement> V | A

Usage: Query only

CHANnel<m>:DATA:YORigin?
CHANnel<m>:DATA:ENVelope:YORigin?
Return the value for binary value 0 of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Yorigin> V | A

Usage: Query only

CHANnel<m>:DATA:YRESolution?
CHANnel<m>:DATA:ENVelope:YRESolution?
Return the vertical bit resolution of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Yresolution> For default waveforms, the resolution is 8 bit.

Usage: Query only

CHANnel<m>:DATA:POINts <Points>
As a setting, the command selects a range of samples. As a query, it returns the number of samples for the selected range.

To get correct results with settings MAX and DMAX, the acquisition must not run when the command is used. To acquire consistent and valid data, use the SINGle command before. Do not cancel a running acquisition with STOP because the waveform data might be incomplete or incorrect.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Points> Number of data points in the selected range.
Default unit:Samples

Setting Parameters:

<Points> DEFault | MAXimum | DMAXimum

Sets the range for the query.

DEFault
Waveform samples displayed on the screen.

MAXimum
Range is the complete memory (only available in STOP mode).

DMAXimum
Display maximum:number of samples in the current waveform record (only available in STOP mode).

Example:

The instrument is set up with an automatic waveform rate. The acquisition will record 1MByte. 800.000 points of 1MByte can be shown within the display. To increase the performance of the device the record is decimated to 6000 points within the displayed area. Those 6000 points are the displayed waveform.

Reading default data will return the decimated waveform of the display.
Use this by setting `CHAN:DATA:POIN DEF`.
A query will return 6000 points.

To read all recorded points within the display area use `CHAN:DATA:POIN DMAX`.
In this case you will get 800.000 points.

Use `CHAN:DATA:POIN MAX` to get the full record including the data points that can not be shown on the display.

FORMat[:DATA] <DataFormat>,<Accuracy>
Defines the format for data export with:
CHANnel<m>:DATA? on page 46
CHANnel<m>:DATA:ENVelope? on page 47
CALCulate:MATH<m>:DATA? on page 96
REFCurve<m>:DATA? on page 102
CALCulate:QMATH:DATA? on page xxx

Parameters:

<DataFormat> ASCii | REAL | UINTeger

ASCii
List of values, for example, 1.23,1.22,1.24,..
<Accuracy> is 0, which means that the instrument selects the number of digits to be returned.
The query returns `ASC,0`.

REAL
Block data, 32 bit float values in 4 byte blocks.
<Accuracy> is always 32. The query returns `REAL,32`.

UINTeger
Unsigned integer format, binary values with length 8 or 16 bit (UINT,8 or UINT,16).
The data range for `UINT,8` is 0 to 255, the data range for `UINT,16` is 0 to 65.535.
For data conversion, you need the results of `...:DATA:XORigin?, ...:DATA:XINCrement?, ...:DATA:Yorigin?,...:DATA:YINCrement?` and `...:DATA:YRESolution?` commands.

*RST: ASC

<Accuracy> 0 | 8 | 16 | 32
Length of a data value in bit

*RST: 0

Example: Set the ASCII data format:

```
FORM ASC
```

Example: Query for data format:

```
FORM?
-> ASC,0
```

FORMat:BORDer <ByteOrder>
Defines the byte order for binary data export if FORMat[:DATA] is set to REAL or UINT.

Parameters:

<ByteOrder> MSBFirst | LSBFirst

MSBFirst
Big endian, most significant byte first, for example:
abcd,acbe,abcf, ...

LSBFirst
Little endian, least significant byte first, for example:
dcba,ebca,fcba, ...

*RST: MSBF

2.2.7 Probes

PROBe<m>:SETup:TYPE?
Queries the type of the probe.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<Type> NONE | ACTive | PASSive

NONE not detected
ACTive active probe
PASSive passive probe

Usage: Query only

PROBe<m>:SETup:ATTenuation:UNIT <Unit>
Selects the unit that the probe can measure.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<Unit> V | A

PROBe<m>:SETup:ATTenuation:MANual <ManualAttenuation>
Sets the attenuation of the probe if the probe was not detected by the instrument.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Parameters:
<ManualAttenuation> Range: 0.001 to 1000

*RST:1

PROBe<m>:SETup:ATTenuation[:AUTO]?
Returns the attenuation of an automatically detected probe.

Suffix:
<m> Selects the input channel. The number of channels depends on the instrument.

Return values:
<ProbeAttenuation> Range: 0.001 to 1000

Usage: Query only

2.3 Trigger

2.3.1 General A Trigger Settings

TRIGger:A:MODE <TriggerMode>
Sets the trigger mode. The trigger mode determines the behaviour of the instrument if no trigger occurs.

Parameters:
<TriggerMode> AUTO | NORMal

AUTO
The instrument triggers repeatedly after a time interval if the trigger conditions are not fulfilled. If a real trigger occurs, it takes precedence.

NORMal
The instrument acquires a waveform only if a trigger occurs.

*RST: AUTO

TRIGger:A:LEVel<n>**[:VALue]** <Level>
Sets the trigger treshold voltage for all A trigger types that require a trigger level.

Suffix:
<n> Selects the trigger input. 1...4 select the corresponding channel, 5 is the external trigger input. The number of channels depends on the instrument.

Parameters:
<Level> Default unit: V

TRIGger:A:SOURce <Source>
Sets the trigger source for the selected A trigger type.

Parameters:

<Source> CH1 | CH2 | CH3 | CH4 | D0...D15 | BUS1 | BUS2 | EXTernanalog | LINE | NONe | PATTern

CH1 | CH2 | CH3 | CH4
One of the input channels is the trigger source. Available channels depend on the instrument type.

D0...D15
Slope trigger on the POD bit. Available digital channels depend on the instrument type.

BUS1 | BUS2
Trigger on serial BUS (depend on the instrument type).

EXTernanalog
External Trigger Input on the rear / front panel.

LINE
AC line for the edge trigger.

NONE
No trigger (e.g. in roll mode).

PATTern
Pattern, if logic trigger is active.

TRIGger:EXTern:COUPling <ExternCoupling>
Sets the coupling for the external trigger input. The command is relevant if `TRIGger:B:SOURce` is set to `EXTernanalog`.

Parameters:

<ExternCoupling> AC | DC

*RST: AC

TRIGger:A:TYPE <Type>
Sets the trigger type for the A trigger.

Parameters:

<Type> EDGE | WIDTh | TV | LOGic | BUS

EDGE: edge trigger
WIDTh: width trigger
TV: video trigger
LOGic: logic trigger
BUS: bus trigger, requires options for serial trigger and decode

2.3.2 Edge Trigger

TRIGger:A:EDGE:SLOPe <Slope>
Sets the slope for the edge trigger (A trigger).

Parameters:
<Slope> POSitive | NEGative | EITHer

POSitive
Rising edge, a positive voltage change

NEGative
Falling edge, a negative voltage change

EITHer
Rising as well as the falling edge (alternating)

*RST: POSitive

TRIGger:A:EDGE:COUPling <Coupling>
Sets the coupling for the trigger source.

Parameters:
<Coupling> DC | AC | HF | ALEVel

DC
Direct Current coupling. The trigger signal remains unchanged.

AC
Alternating Current coupling. A 5 Hz high pass filter removes the DC offset voltage from the trigger signal.

HF
High frequency coupling. A 15 kHz high-pass filter removes lower frequencies from the trigger signal. Use this mode only with very high frequency signals.

ALEVel
Auto Level in HMO series 722 to 2024.

*RST: DC

TRIGger:A:EDGE:FILTer:LPASs <State>
Turns an additional 5 kHz low-pass filter in the trigger path on or off. This filter removes higher frequencies and is available with AC and DC coupling.

Parameters:
<State> ON | OFF

*RST: OFF

TRIGger:A:EDGE:FILTer:NREJect <State>
Turns an additional 100 MHz low-pass filter in the trigger path on or off. This filter removes higher frequencies and is available with AC and DC coupling.

Parameters:
<State> ON | OFF

*RST: OFF

2.3.3 Width Trigger

TRIGger:A:WIDTh:POLarity <Polarity>
Sets the polarity of the pulse.

Parameters:
<Polarity> POSitive | NEGative

POSitive
Positive going pulse, the width is defined from the rising to the falling slopes.

NEGative
Negative going pulse, the width is defined from the falling to the rising slopes.

*RST: POSitive

TRIGger:A:WIDTh:RANGe <RangeMode>
Defines how the measured pulse width is compared with the given limit(s).

Parameters:
<RangeMode> WITHin | OUTSide | SHORter | LONGer

WITHin | OUTSide
Triggers on pulses inside or outside a range defined by time ± delta. The time is specified with `TRIGger:A:WIDTh:WIDTh`, the range around is defined with `TRIGger:A:WIDTh:DELTa`.

SHORter | LONGer
Triggers on pulses shorter or longer than a time set with `TRIGger:A:WIDTh:WIDTh`.

*RST: LONGer

TRIGger:A:WIDTh:DELTa <Delta>
Defines a range around the width value specified using TRIGger:A:WIDTh:WIDTh.

Parameters:
<Delta>
Range ±Δt („Variation")
Range: 3.2E-9 to maximum value depends on the defined pulse width
Increment: 3.2E-9

*RST: 3.2E-9

TRIGger:A:WIDTh:WIDTh <Time1>
For the ranges WITHin and OUTSide (defined using `TRIGger:A:WIDTh:RANGe`), the <Time1> defines the center of a range which is defined by the limits ±<Delta> (set with `TRIGger:A:WIDTh:DELTa`).

For the ranges SHORter and LONGer, the width defines the maximum and minimum pulse width, respectively.

Parameters:
<Time1> Center value, maximum value or minimum value depending on the defined range type.

Range: 19.2E-9 to 107.374E-3 (depending on model)
Increment: 6.4E-9, for t > 1 ms:1 µs (depending on model)

*RST: 19.2E-9

2.3.4 Video/TV Trigger

TRIGger:A:TV:STANdard <Standard>
Selects the color television standard.

Parameters:
<Standard> PAL | NTSC | SECam | PALM | I576 | P720 | P1080 | I1080

*RST: PAL

TRIGger:A:TV:POLarity <Polarity>
Sets the polarity of the sync pulses. The edges of the sync pulses are used for triggering.

Parameters:
<Polarity> POSitive | NEGative

POSitive If the video modulation is positve, the sync pulses are negative.
NEGative If the modulation is negative, sync pulses are positive.

*RST: NEGative

TRIGger:A:TV:FIELd <Field>
Sets the trigger on the beginning of the video signal fields, or on the beginning of video signal lines.

Parameters:
<Field> EVEN | ODD | ALL | LINE | ALINe

EVEN Triggers only on even half frames.
ODD Triggers only on odd half frames.
ALL Triggers on all frames.
LINE Triggers on the beginning of a specified line in any field. The line number is set with `TRIGger:A:TV:LINE.`
ALINe Triggers on the beginning of all video signal lines.

*RST: ALL

TRIGger:A:TV:LINE <Line>
Sets an exact line number if `TRIGger:A:TV:FIELd` is set to LINE.

Parameters:
<Line> Range: 1 to 525, 625 , ... , 1080 depending on the TV standard
Increment: 1

*RST: 1

2.3.5 Pattern Trigger

TRIGger:A:PATTern:SOURce <SourceString>
Select the state for each digital channel.

Parameters:

<SourceString> string containing 0, 1, or X for each channel

1: high, the signal voltage is higher than the trigger level.
0: low, the signal voltage is lower than the trigger level.
X: Don't care. the channel does not affect the trigger.

Example: TRIG:A:PATT:SOUR „1X10"

CH1, CH3, and NOT CH4 are logically combined with
`TRIGger:A:PATTern:FUNCtion`, CH2 does not matter (don't care).

TRIGger:A:PATTern:FUNCtion <Function>
Sets the logical combination of the trigger states of the channels.

Parameters:

<Function> AND | OR

AND
The required states of all channels must appear in the input signal at the same time.

OR
At least one of the channels must have the required state.

*RST: AND

TRIGger:A:PATTern:CONDition <Condition>
Sets the trigger point depending on the result of the logical combination of the channel states.

Parameters:

<Condition> "TRUE" | "FALSE"

*RST: "TRUE"

TRIGger:A:PATTern:MODE <PatternMode>
Sets the duration function of the pattern trigger.

Parameters:
<PatternMode> OFF | TIMeout | WIDTh

OFF
only pattern trigger without duration

TIMeout
pattern trigger with duration and timeout

WIDTh
using duration and comparision of the width of the logical combined channel

*RST: OFF

TRIGger:A:PATTern:WIDTh[:WIDTh] <numeric_value>

For the ranges WITHin and OUTSide (defined using `TRIGger:A:PATTern:WIDTh:RANGe`), the <numeric_value> defines the center of a range which is defined by the limits ±<Delta> (set with `TRIGger:A:PATTern:WIDTh:DELTa`).

For the ranges SHORter and LONGer, the width defines the maximum and minimum pulse width, respectively.

Parameters:
<numeric_value> Center value, maximum value or minimum value depending on the defined range type.

Range: 19.2E-9 to 107.374E-3 (depending on model)
Increment: 6.4E-9, for t > 1 ms:1 µs

*RST: 19.2E-9

TRIGger:A:PATTern:WIDTh:DELTa <PatternDelta>
Defines a range around the width value specified using `TRIGger:A:PATTern:WIDTh[:WIDTh]`

Parameters:
<Delta> Range ±Δt („Variation")
Range: 3.2E-9 to maximum value depends on the defined pulse width
Increment: 3.2E-9

*RST: 3.2E-9

TRIGger:A:PATTern:WIDTh:RANGe <PatternRange>
Defines a range around the width value specified using TRIGger:A:PATTern:WIDTh:WIDTh.

Parameters:
<PatternRange> WITHin | OUTSide | SHORter | LONGer

WITHin | OUTSide
Triggers on pulses inside or outside a range defined by time ± delta. The time is specified with `TRIGger:A:PATTern:WIDTh[:WIDTh]`, the range around is defined with `TRIGger:A:PATTern:WIDTh:DELTa`.

SHORter | LONGer
Triggers on pulses shorter or longer than a time set with `TRIGger:A:PATTern:WIDTh[:WIDTh]`

*RST: LONGer

2.3.6 B-Trigger

TRIGger:B:ENABle <State>
Activates or deactivates the second trigger.
The instrument triggers if both trigger event conditions (A and B) are fulfilled.

Parameters:
<State> ON | OFF

*RST: OFF

TRIGger:B:SOURce <Source>

Selects one of the input channels as B-trigger source. Available channels depend on the instrument type.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4

*RST: CH1

TRIGger:B:EDGE:SLOPe <Slope>
Sets the edge for the B-trigger.

Parameters:
<Slope> POSitive | NEGative | EITHer (alternating)

*RST: POSitive

TRIGger:B:LEVel <Level>
Sets the trigger level for the B-trigger event.

Parameters:
<Level> Default unit: V

*RST: 0

TRIGger:B:MODE <Mode>
Defines the delay type of the B-trigger.

Parameters:
<Mode> DELay | EVENts

DELay
Time delay, set with `TRIGger:B:DELay`

EVENts
Event count delay, set with `TRIGger:B:EVENt:COUNt`

*RST: DELay

TRIGger:B:DELay <DelayTime>
Sets the time the instrument waits after an A-event until it recognizes B-events.
Before setting the delay time, `TRIGger:B:MODE` must be set to DELay.

Parameters:

<DelayTime>	Range:	8E-9 min., depending on model
	Increment:	depending on model
	Default unit:	s
	*RST:	8E-9 min., depending on model

TRIGger:B:EVENt:COUNt <EventCnt>
Sets a number of B-trigger events that fulfill all B-trigger conditions but do not cause the trigger. The oscilloscope triggers on the n-th event (the last of the specified number of events).
Before setting the event number, `TRIGger:B:MODE` must be set to EVENts.

Parameters:

<EventCnt>	Number of B-events	
	Range:	1 to 65535
	Increment:	1
	*RST:	1

2.4 Display

2.4.1 Basic Display Settings

This chapter describes commands that configure the screen display.

General Display Settings:

XYZ-Setup:

Intensities

Waveform, Auxilary Cursors and Grid Settings

DISPlay:MODE <Mode>
Sets the diagram mode.

Parameters:

<Mode> YT | XY

YT
Default time diagram with a time axis in x-direction and the signal amplitudes displayed in y-direction.

XY
XY-diagram, combines the voltage levels of two waveforms in one diagram.

*RST: YT

DISPlay:PALette <Palette>
Sets the color and brightness of the displayed waveform samples depending on their cumulative occurance.

Parameters:

<Palette> NORMal | INVerse | FCOLor | IFColor

NORMal
Values that occur frequently are brighter than rare values.

INVerse
Rare values are brighter than frequent values, inverse to the NORMal brightness.

FColor
Rare values are displayed in blue, while more frequent values are red and very frequent values are displayed in yellow or white, with various colors inbetween.

IFColor
Inverses the FColor setting:rare values are yellow or white while frequent values are blue.

*RST: NORMal

DISPlay:VSCREEN:ENABle <Enable>
Switches the virtual screen on or off.

Parameters:
<Enable> ON | OFF

*RST: OFF

DISPlay:VSCREEN:POSition <Position>
Set the position of the virtual screen window.

Parameters:
<Position> Range: 2 to -8
Increment: 0.02
Default unit: div

*RST: 0

DISPlay:XY:XSOurce <Source>
Defines the source to be displayed in x direction in an XY-diagram, replacing the usual time base.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4

*RST: CH1

DISPlay:XY:Y1Source <Source>
Defines the y source of the first xy waveform in the diagram.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4

*RST: CH2

DISPlay:XY:Y2Source <Source>
Defines the y source of the second xy waveform in the diagram.

Parameters:
<Source> NONE | CH1 | CH2 | CH3 | CH4

*RST: NONE

DISPlay:XY:ZMODe <Mode>
Activates or deactivates the intensity control of the waveform via an additional signal source and sets the intensity mode.

Parameters:

<Mode> ANALog | DIGital | OFF

ANALog
Modulated intensity; Intensity is modulated continuously according to the selected source Z.

DIGital
Intensity is determined by a threshold value defined with `DISPlay:XY:ZTHReshold`. If the Z signal value is below the selected threshold, the corresponding x/y point is displayed with lowest intensity. If the Z signal value is above the threshold, the x/y point is displayed with the defined intensity level.

OFF
Intensity control is deactivated.

*RST: OFF

DISPlay:XY:ZTHReshold <Zthreshold>
Defines the threshold for intensity with a two-state modulation, if `DISPlay:XY:ZMODe` is set to DIGital.

Parameters:

<Zthreshold> Threshold for visibility on the screen
Range: -10 to 10
Increment: depends on the scaling of the channel that is assigned to Z
Default unit: V

*RST: 0

DISPlay:XY:ZSOurce <Source>
Defines the source to be used to determine the intensity of the xy-waveform.

Parameters:

<Source> CH1 | CH2 | CH3 | CH4

*RST: CH1

DISPlay:INTensity:WAVeform <Intensity>
Defines the strength of the waveform line in the diagram.

Parameters:

<Intensity> Value in percent
Range: 0 to 100
Increment: 1
Default unit: %

*RST: not available, (*RST does not change the intensity)

DISPlay:INTensity:BACKlight <Intensity>
Defines the intensity of the background lighting of the display.

Parameters:

<Intensity> Value in percent
Range: 10 to 100
Increment: 1
Default unit: %

*RST: not available, (*RST does not change the intensity)

DISPlay:INTensity:GRID <Intensity>
Defines the intensity of the grid on the screen.

Parameters:

<Intensity> Value in percent
Range: 0 to 100
Default unit: %

*RST: not available, (*RST does not change the intensity)

DISPlay:PERSistence:STATe <State>
Defines whether the waveform persists on the screen or whether the screen is refreshed continuously.

Parameters:

<State> ON | OFF

ON
The waveform persists for the time defined using `DISPlay:PERSistence:TIME`.

OFF
The waveform does not persist on the screen. Only the currently measured values are displayed at any time.

*RST: OFF

DISPlay:PERSistence:TIME <Time>
Persistence time if persistence is active (see `DISPlay:PERSistence:STATe`).

Each new data point in the diagram area remains on the screen for the duration defined here. To set infinite persistence, use `DISPlay:PERSistence:INFinite`.

Parameters:

<Time> Range: 50E-3 to infinite
Increment: min. 50E-3, increasing increment with increasing persistence time
Default unit: s

*RST: 50E-3

DISPlay:PERSistence:INFinite <InfPersistence>
Sets the persistence time to infinite if `DISPlay:PERSistence:STATe` is ON. each new data point remains on the screen infinitely until this setting is changed or the persistence is cleared.

Parameters:

<InfPersistence> ON | OFF

*RST: OFF

DISPlay:PERSistence:TIME:AUTO <Auto>
The optimal persistence time is determined automatically by the instrument.

Parameters:
<Auto> ON | OFF

DISPlay:PERSistence:CLEar
Removes the displayed persistent waveform from the screen.

Usage: Event

DISPlay:STYLe <Style>
Defines how the waveform data is displayed

Parameters:
<Style> VECTors | DOTS

VECTors: Individual data points are connected by a line.
DOTS: Only the data points are displayed.

*RST: VECT

DISPlay:GRID:STYLe <Style>
Defines how the grid is displayed.

Parameters:
<Style> LINes | RETicle | NONE

LINes: Displays the grid as horizontal and vertical lines.
RETicle: Displays crosshairs instead of a grid.
NONE: No grid is displayed.

*RST: LIN

2.4.2 Zoom

TIMebase:ZOOM:STATe <ZoomState>
Switches the zoom window on or off.

Parameters:
<ZoomState> ON | OFF

*RST: OFF

TIMebase:ZOOM:SCALe <ZoomScale>
Defines the time base in the zoom diagram in seconds per division.

Parameters:
<ZoomScale> Scaling of the zoom time base
Default unit: s/div

*RST: 50e-6

TIMebase:ZOOM:TIME <Time>
Defines the offset of the trigger point to the reference point of the zoom diagram.

Parameters:
<Time> Default unit: s

*RST: 0

TIMebase:ZOOM:POSition <Position>
Defines the position of the zoom reference point (the reference point of the zoom window) in relation to the reference point of original time base.

Parameters:
<Position> Range: 0 to 100, depending on the zoom time base
Default unit: %

*RST: 50

2.4.3 Markers (Timestamps)

TSTamp:SET
Sets a new marker (timestamp) at the reference point of the display, unless an existing marker is already set there. The reference point is set with „TIMebase:REFerence“ on page 33.

Usage: Event

TSTamp:NEXT
Moves the next marker (timestamp, to the right) to the reference point of the display or zoom area.

Usage: Event

TSTamp:PREVious
Moves the previous marker (timestamp, to the left) to the reference point of the display or zoom area.

Usage: Event

TSTamp:CLEar
Deletes the marker (timestamp) at the reference point. The reference point is set with „TIMebase:REFerence“ on page 33.

Usage: Event

TSTamp:ACLear
Deletes all markers (timestamps).

Usage: Event

2.5 Measurements

This chapter describes functions that configure or perform cursor and automatic measurements.

2.5.1 Cursor

CURSor<m>:AOFF
Switches the cursor off.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Usage: Event

CURSor<m>:STATe <State>
Activates or deactivates the cursor measurement.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<State> ON | OFF

*RST: OFF

CURSor<**m**>**:FUNCtion** <Type>
Defines the cursor measurement type.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Type> HORizontal | VERTical | PAIRed | HRATio | VRATio | PPCount | NPCount | RECount | FECount | MEAN | RMS | RTIMe | FTIMe | PEAK | UPEakvalue | LPEakvalue | STDDEV | PDCYcle | NDCYcle

HORizontal: Sets two horizontal cursor lines and measures the voltages at the two cursor positions and the delta of the two values.
V**ERTical:** Sets two vertical cursor lines and measures the time from the trigger point to each cursor, the time between the two cursors and the frequency calculated from that time.
PAIRed: V-Marker
HRATio: Ratio of the x-values (e.g. a duty cycle) between the first and second cursors and the first and third cursors
VRATio: Ratio of the y-values (e.g. overshooting) between the first and second cursors and the first and third cursors
PPCount: Count positive pulses
NPCount: Count negative pulses
RECount: Count rising edges
FECount: Count falling edges
MEAN: Mean value
RMS: Root mean square
RTIMe: Rise time, tr
FTIMe: Fall time, tf. The reference level for rise and fall time measurement is set with „REFLevel<m>:RELative:MODE“ on page 78.
PEAK: Absolute difference between the two peak values, Vpp
UPEakvalue: Upper peak value, Vp+
LPEakvalue: Lower peak value, Vp
STDDEV standard deviation
PDCYcle positive duty cycle
NDCYcle negative duty cycle

*RST: PAIR

CURSor<**m**>**:SOURce** <Source>
Defines the source of the cursor measurement as one of the active signal, reference or math channels.

Suffix:
<m> 1 (the numeric suffix is irrelevant)
Parameters:
<Source> NONE | CH1 | CH2 | CH3 | CH4 | POD1 | POD2 | D0...D15 |QMA | MA1 | MA2 | MA3 | MA4 | MA5 | RE1 | RE2 | RE3 | RE4 | XY1 | XY2

CH1 | CH2 | CH3 | CH4
Active signal channels 1 to 4

POD1 | POD2
Active logic POD (depending on the instrument type)

D0...D15
Active digital channels from D0 up to D15 (depending on the instrument type)

QMA
Active quick math channel

MA1 | MA2 | MA3 | MA4 | MA5
Active math channels 1 to 5

RE1 | RE2 | RE3 | RE4
Active reference channels 1 to 4

XY1 | XY2
Active XY-waveform

*RST: CH1

CURSor<m>:TRACking[:STATe] <State>
If set to ON, the V-Marker cursor measurement is enabled.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<State> ON | OFF

*RST: OFF

CURSor<m>:X1Position <Xposition1>
Specifies the position of the first cursor on the time axis.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Xposition1> Default unit: s

CURSor<m>:X2Position <Xposition2>
Specifies the position of the second cursor on the time axis.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Xposition2> Default unit: s

CURSor<m>:X3Position <Xposition3>
Specifies the position of the third cursor on the time axis for the Ratio X measurement.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Xposition3> Default unit: s

CURSor<m>:Y1Position <Yposition1>
Specifies the position of the first horizontal cursor on the y-axis.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Yposition1> Default unit: V

CURSor<**m**>**:Y2Position** <Yposition2>
Specifies the position of the second horizontal cursor on the y-axis.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Yposition2> Default unit: V

CURSor<**m**>**:Y3Position** <Yposition3>
Specifies the position of the third horizontal cursor on the y-axis for Ratio Y measurements.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Yposition3> Default unit: V

CURSor<**m**>**:TRACking:SCALe[:STATe]** <State>
Enables the adjustment of cursor lines if the vertical or horizontal scales are changed.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<State> ON | OFF

ON
Cursor lines keep their relative position to the waveform.

OFF
Cursor lines remain on their position on the display if the scaling is changed.

*RST: OFF

CURSor<**m**>**:YCOupling** <Coupling>
CURSor<**m**>**:XCOupling** <Coupling>
If enabled, the cursors of a set are coupled so that the distance between the two remains the same if one cursor is moved.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Coupling> ON | OFF

*RST: OFF

CURSor<m>:RESult?
Returns a result depending on the selected cursor function.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Return values:
<Value> Measurement result

Example: If `CURSor<m>:FUNCtion` is set to RTIM, `CURSor<m>:RESult?` will return the risetime of a slope between both cursors.
If the function is set to MEAN, `CURSor<m>:RESult?` will return the measured mean value.
If there is no specific value, `CURSor<m>:RESult?` will return the position of the first cursor.

Usage: Query only

CURSor<m>:XDELta:INVerse?
Returns the inverse time difference between the two cursors (1/Δt).

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Return values:
<DeltaInverse> Default unit: 1/s

Usage: Query only

CURSor<m>:XDELta[:VALue]?
Returns the time difference between the two cursors (Δt).

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Return values:
<Delta> Default unit: s

Usage: Query only

CURSor<m>:YDELta:SLOPe?
Returns the inverse value of the voltage difference - the reciprocal of the vertical distance of two horizontal cursor lines:1/ΔV.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Return values:
<DeltyYslope> Inverse value

Usage: Query only

CURSor<m>:YDELta[:VALue]?
Queries the delta of the values in y-direction at the two cursors.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Return values:
<DeltaY> Delta value in V

Usage: Query only

CURsor<m>:XRATio:UNIT <Unit>
Sets the unit for X Ratio measurements with `CURSor<m>:XRATio[:VALue]`.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Unit> RATio | PCT | GRD | PI

RATio: floating value
PCT: percent
GRD: degree
PI: radian

*RST: RAT

CURSor<m>:XRATio[:VALue]?
Returns the ratio of the x-values (e.g. a duty cycle) between the first and second cursors and the first and third cursors:(x2-x1)/(x3-x1).

Set the unit of the result with `CURsor<m>:XRATio:UNIT`.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Return values:
<Ratio> Numeric value corresponding to the specified unit.

Usage: Query only

CURSor<m>:YRATio:UNIT <Unit>
Sets the unit for Y Ratio measurements with `CURSor<m>:YRATio[:VALue]`.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Parameters:
<Unit> RATio | PCT

RATio: floating value
PCT: percent

*RST: RAT

CURSor<m>:YRATio[:VALue]?
Provides three cursors and measures the ratio of the y-values (e.g. overshooting) between the first and second cursors and the first and third cursors:(y2-y1)/(y3-y1).
Set the unit of the result with `CURSor<m>:YRATio:UNIT`.

Suffix:
<m> 1 (the numeric suffix is irrelevant)

Return values:
<Ratio> Numeric value corresponding to the specified unit.

Usage: Query only

2.5.2 Automatic Measurements

MEASurement<m>:AOFF
Stops the quick measurement.

Suffix:
<m> 1...6
The numeric suffix is irrelevant.

Usage: Event

MEASurement<m>:AON
Starts the quick measurement.

Suffix:
<m> 1...6
The numeric suffix is irrelevant.

Usage: Event

MEASurement<m>:ARESult?
Returns the results of the quick measurement.

Suffix:
<m> 1...6
Selects the measurement.

Return values:
<QuickMeasData> List of values
Quick measurement results are listed in the following order:
PEAK, UPE, LPE, CYCR, CYCM, PER, FREQ, RTIM , FTIM

Usage: Query only

MEASurement<m>[:ENABle] <State>
Activates or deactivates the selected measurement (1-6). Only the results of active measurements are displayed on the screen.

Suffix:
<m> 1...6
Selects the measurement.

Parameters:
<State> ON | OFF

*RST: OFF

MEASurement<m>:SOURce <SignalSource>[,<ReferenceSource>]
Selects one of the active signal, reference or math channels as the source of the selected measurement.

Suffix:
<m> 1...6

Parameters:
<SignalSource> NONE | CH1 | CH2 | CH3 | CH4 | D0...D15 | QMA | MA1 | MA2 | MA3 | MA4 | MA5 | RE1 | RE2 | RE3 | RE4

<ReferenceSource> NONE | CH1 | CH2 | CH3 | CH4 | MA1 | MA2 | MA3 | MA4 | MA5 | RE1 | RE2 | RE3 | RE4

CH1 | CH2 | CH3 | CH4
Active signal channels 1 to 4

D0...D15
Active digital channels from D0 up to D15 (depending on the instrument type)

QMA
Active quick math channel

MA1 | MA2 | MA3 | MA4 | MA5
Active math channels 1 to 5

RE1 | RE2 | RE3 | RE4
Active reference channels 1 to 4

*RST: CH1

MEASurement<m>:CATegory?
Returns the measurement category. Currently, the instrument supports only yt-measurements.

Suffix:
<m> 1...6
Selects the measurement.

Return values:
<Category> AMPTime

AMPtime: yt-measurements

*RST: AMPT

Usage: Query only

REFLevel<m>:RELative:MODE <RelativeMode>
Sets the lower and upper reference levels for rise and fall time mesurements (cursor and automatic mesurements), defined as percentages of the high signal level.

Suffix:
<m> 1...7
Measurement to which the reference level belongs.
Suffix 1...6 assign the measurement places 1 to 6 (auto measure).
Suffix 7 assigns the cursor mesurement.

Parameters:
<RelativeMode> TEN | TWENty

TEN: 10/90%
TWENty: 20/80%

Example:
```
REFL2:REL:MODE TWENty
MEAS2:MAIN RTIM
```
Sets the reference level for measurement place 2 and measures the rise time between these levels:
lower reference level = 20% of high signal level
upper reference level = 80% of high signal level

*RST: TEN

MEASurement<m>:MAIN <MeasType>
Defines the measurement type to be performed on the selected source. To query the results, use
„CALCulate:MATH<m>:DATA?" on page 96.

Suffix:
<m> 1...6
Selects the measurement type.

Parameters:
<MeasType> FREQuency | PERiod | PEAK | UPEakvalue | LPEakvalue | PPCount | NPCount | RECount | FECount | HIGH | LOW | AMPLitude | MEAN | RMS | RTIMe | FTIMe | PDCYcle | NDCYcle | PPWidth | NPWidth | CYCMean | CYCRms | STDDev | TFRequency | TPERiode | POVershoot | NOVershoot | DELay | PHASe

FREQuency
Frequency of the signal. The result is based on the length of the left-most signal period within the displayed section of the waveform of the selected channel.

PERiod
Length of the left-most signal period within the displayed section of the waveform of the selected channel.

PEAK
Peak-to-peak value within the displayed section of the waveform of the selected channel.

UPEakvalue
Maximum value within the displayed section of the waveform of the selected channel.

LPEakvalue
Minimum value within the displayed section of the waveform of the selected channel.

PPCount
Counts positive pulses.

NPCount
Counts negative pulses.

RECount
Counts the number of rising edges.

FECount
Counts the number of falling edges.

HIGH
Mean value of the high level of a square wave.

LOW
Mean value of the low level of a square wave.

AMPLitude
Amplitude of a square wave.

MEAN
Mean value of the complete displayed waveform of the selected channel.

RMS
RMS (Root Mean Square) value of the voltage of the complete displayed waveform of the selected channel.

RTIMe | FTIMe
Rise or falling time of the left-most rising edge within the displayed section of the waveform of the selected channel. The reference level for this mesurement is set with „REFLevel<m>:RELative:MODE“ on page 78.

PDCycle | NDCycle
Measure the positive or negative duty cycle.

PPWidth | NPWidth
Measure the width of positive or negative pulses.

CYCMean
Mean value of the left-most signal period of the waveform of the selected channel.

CYCRms
RMS (Root Mean Square) value of the voltage of the left-most signal period of the waveform of the selected channel.

STDDev
Measures the standard deviation of the waveform.

TFRequency
Measures the frequency of the trigger signal based on the length of its period.

TPERiode
Measures the length of the trigger signal periods (hardware counter).

POVershoot
Measures the positive overshoot of the waveform.

NOVershoot
Measures the negative overshoot (undershoot) of the waveform.

Delay
Time difference between two edges of the same or different waveforms.
The waveforms are selected with MEASurement<m>:SOURce, and the edges with MEASurement<m>:DELay:SLOPe.

Phase
Phase difference between two waveforms [(time difference/period) x 360].
The waveforms are selected with MEASurement<m>:SOURce.

*RST: NONE (measurement is off)

MEASurement<m>:RESult? [<MeasType>]
Returns the result of the specified measurement type.

Suffix:
<m> 1...6
Selects the measurement type.

Return values:
<Value> Measurement result

Query Parameters:
<MeasType> FREQuency | PERiod | PEAK | UPEakvalue | LPEakvalue | PPCount | NPCount | RECount | FECount | HIGH | LOW | AMPLitude | MEAN | RMS | RTIMe | FTIMe PDCYcle | NDCYcle | PPWidth | NPWidth | CYCMean | CYCRms | STDDev | TFRequency | TPERiode | POVershoot | NOVershoot

Specifies the measurement type. See „MEASurement<m>:MAIN“ on page 78.

Usage: Query only

MEASurement<m>:RESult:AVG? [<AverageValue>]
Returns the average value of the current mesurement series. The number of waveforms used for calculation is defined with „MEASurement<m>:STATistics:WEIGht“ on page 82.

Suffix:
<m> 1...6
Selects the measurement type.

Query parameters:
<AverageValue> Statistic value

Usage: Query only

MEASurement<m>:RESult:STDDev? [<StandardDeviation>]
Returns the statistical standard deviation of the current mesurement series. The number of waveforms used for calculation is defined with „MEASurement<m>:STATistics:WEIGht“ on page 82.

Suffix:
<m> 1...6
Selects the measurement type.

Query parameters:
<StandardDeviation> Statistic value

Usage: Query only

MEASurement<m>:RESult:NPEak? [<NegativePeak>]
Returns the minimum measurement value of the current measurement series.

Suffix:
<m> 1...6
Selects the measurement type.

Query parameters:
<NegativePeak> Minimum measurement value

Usage: Query only

MEASurement<m>:RESult:PPEak? [<PositivePeak>]
Returns the maximum measurement value of the current measurement series.

Suffix:
<m> 1...6

Selects the measurement type.

Query parameters:
<PositivePeak> Maximum measurement value

Usage: Query only

MEASurement<m>:RESult:WFMCount? [<WaveformCount>]
Returns the current number of measured waveforms.

Suffix:
<m> 1...6
Selects the measurement type.

Query parameters:
<WaveformCount> **Number of measured waveforms**

Usage: **Query only**

MEASurement<m>:DELay:SLOPe <SignalSlope>,<ReferenceSlope>
Sets the edges to be used for delay measurement. The associated waveforms are defined with MEASurement<m>:SOURce

Parameters:

<SignalSlope> POSitive | NEGative
Slope of source 1 (first waveform)

<ReferenceSlope> POSitive | NEGative
Slope of source 2 (second waveform)

*RST: POS

MEASurement<m>:STATistics[:ENABle] <StatisticEnable>
Activates or deactivates the statistical evaluation for the selected measurement.

Suffix:

<m> 1...6
Selects the measurement type.

Parameters:

<StatisticEnable> ON | OFF

*RST: OFF

MEASurement<m>:STATistics:RESet
Deletes the statistical results for the selected measurement, and starts a new statistical evaluation, if the acquisition is running. The waveform count is set to 0 and all measurement values are set to NAN.

Suffix:

<m> 1...6
Selects the measurement type.

Usage: Event

MEASurement<m>:STATistics:WEIGht <AverageCount>
Sets the number of measured waveforms used for calculation of average and standard deviation.

Suffix:

<m> 1...6
Selects the measurement type.

Parameters:

<AverageCount> Statistics average count
Range: 2 to 1000
Increment: 1

*RST: 1000

2.6 Search functions

2.6.1 Search

SEARch:STATe <SearchState>
Enable or disable the search function

Parameters:
<SearchState> ON | OFF

*RST: OFF

SEARch:SOURce <SearchSource>
Set the source for the search function.

Parameters:
<SearchSource> CH1 | CH2 | CH3 | CH4 | QMA | MA1 | MA2 | MA3 | MA4 | MA5 | RE1 | RE2 | RE3 | RE4

CH1 | CH2 | CH3 | CH4
Signal channels 1 to 4

QMA
Quick math channel

MA1 | MA2 | MA3 | MA4 | MA5
Math channels 1 to 5

RE1 | RE2 | RE3 | RE4
Reference channels 1 to 4

*RST: CH1

SEARch:CONDition <SearchCondition>
Set the condition for the search function

Parameters:
<SearchCondition> EDGE | WIDTh | PEAK | RUNT | RTIMe

*RST: EDGE

SEARch:RCOunt
Returns the number of events which meet the search condition.

Return values:
<ResultCount> <numeric_value>

*RST: 0

SEARch:RESult<n>?
Returns the result values of the specified search result.

See also: „SEARch:RESult:ALL?" below

Suffix:
<n> *
Number of the search result

Return values:
<Result> Comma-separated value list
Optional value depending on search condition:

EDGE	unused
WIDTh	Puls width
PEAK	peak value of the peak
RUNT	width of the runt
RTIMe	risetime

Example:
```
SEARch:RESult3?
```
Returns the result values of the third search result.
```
3,-4.1660e-04,0,PEAK,NEGATIVE,-1.530e-02
```

Usage: Query only

SEARch:RESult:ALL?
Returns a list of the results separated by comma.

Return values:
<AllResult> List of results separeted by comma

For each result, six values are returned:
1. Result number as indicated in the search results table
2. X-position (time) of the search result
3. Y-position of the search result, currently not relevant
4. Type of the search result (Edge, Peak, ...)
5. Slope or polarity of the search result
6. For peak searches, the value contains the peak voltage. For width searches, it contains the pulse width. For edge searches, the value is not relevant.

Example:
```
SEARch:RESult:ALL?
```
Returns all four results of a peak search:
```
1,-4.7750e-04,0,PEAK,NEGATIVE,-1.530e-02,
2,-4.4630e-04,0,PEAK,NEGATIVE,-1.530e-02,
3,-4.1660e-04,0,PEAK,NEGATIVE,-1.530e-02,
4,-3.8690e-04,0,PEAK,NEGATIVE,-1.530e-02
```

Usage: Query only

SEARch:RESDiagram:SHOW <ResultShow>
Enable or disable list view of search results on the display.

Parameters:
<ResultShow> ON | OFF

*RST:OFF

2.6.2 Peak

SEARch:MEASure:PEAK:POLarity <Polarity>
Set the polarity of the peak

Parameters:
<Polarity> POSitive | NEGative | EITHer

*RST: POS

SEARch:MEASure:LEVEL:PEAK:MAGNitude <Magnitude>
Set the magnitude of the peak

Parameters:
<Magnitude> Default unit:V

2.6.3 Edge

SEARch:TRIGger:EDGE:SLOPe <Slope>
Set the slope of the edge.

Parameters:
<Slope> POSitive | NEGative | EITHer

*RST: POS

SEARch:TRIGger:EDGE:LEVel <Level>
Sets the voltage level for the edge search.

Parameters:
<Level> <numeric_value>

*RST: 500E-3 = 500mV

SEARch:TRIGger:EDGE:LEVel:DELTa <DeltaLevel>
Sets a hysteresis range above and below the search level to avoid unwanted search results caused by noise oscillation around the level.

Parameters:
<DeltaLevel> Range: Lower limit depends on vertical scale and other settings, no upper limit

*RST: 200E-3 = 200mV

2.6.4 Width

SEARch:TRIGger:WIDTh:POLarity <Polarity>
Set the polarity of the pulse.

Parameters:
<Polarity> POSitive | NEGative

*RST: POS

SEARch:TRIGger:WIDTh:LEVel <Level>
Set the level of the pulse

Parameters:
<Level> <numeric_value>

*RST: 500E-6

SEARch:TRIGger:WIDTh:LEVel:DELTa <DeltaLevel>
Set the hysteresis of the level of the pulse.

Parameters:
<DeltaLevel> Range:Lower limit depends on vertical scale and other settings, no upper limit

*RST: 200E-6

SEARch:TRIGger:WIDTh:WIDTh <Width>
Set the width of the pulse.

Parameters:
<Width> Default unit:s

SEARch:TRIGger:WIDTh:DELTa <DeltaWidth>
Sets a range Δt to the reference pulse width set with „SEARch:TRIGger:WIDTh:WIDTh“ on page 86,
if SEARch:TRIGger:WIDTh:RANGe is set to WITHin or OUTSide

Parameters:
<DeltaWidth> Range:Lower limit depends on the resolution, practically no upper limit

SEARch:TRIGger:WIDTh:RANGe <Range>
Set the comparision of the width of the pulse

Parameters:
<Range> WITHin | OUTSide | SHORter | LONGer

WITHin
Finds pulses inside the range width ± Δt.

OUTSide
Finds pulses outside the range width ± Δt.

SHORter
Finds pulses shorter than the given width.

LONGer
Finds pulses longer than the given width.

*RST: WITH

2.6.5 Runt

SEARch:TRIGger:RUNT:POLarity <Polarity>
Set the polarity of the Runt

Parameters:
<Polarity> POSitive | NEGative | EITHer

*RST: POS

SEARch:TRIGger:RUNT:WIDTh <Width>
Sets the reference runt pulse width, the nominal value for comparisons.

Parameters:
<Width> Range: Depends on various settings, mainly time base and sample rate

*RST: 200e-6

SEARch:TRIGger:RUNT:DELTa <DeltaWidth>
Sets a range Δt to the reference pulse width set with „SEARch:TRIGger:RUNT:WIDTh“ on page 87,
if SEARch:TRIGger:RUNT:RANGe is set to WITHin or OUTSide.

Parameters:
<DeltaWidth> Range: Depends on various settings, mainly time base and sample rate

*RST: 50e-6

SEARch:TRIGger:RUNT:RANGe <Range>
Sets how the measured pulse width is compared with the given limit(s).
To set the width, use SEARch:TRIGger:RUNT:WIDTh .
To set the range ± Δt, use SEARch:TRIGger:RUNT:DELTa .

Parameters:
<Range> LONGer | SHORter | WITHin | OUTSide

LONGer
Finds pulses longer than the given width.

SHORter
Finds pulses shorter than the given width.

WITHin
Finds pulses inside the range width ± Δt.

OUTSide
Finds pulses outside the range width ± Δt.

*RST: LONG

SEARch:TRIGger:LEVel:RUNT:LOWer <LowerLevel>
Sets the lower voltage threshold for runt detection. A positive runt crosses the lower level twice without crossing the upper level.

Parameters:
<LowerLevel> Default unit:V

*RST: 400E-3

SEARch:TRIGger:LEVel:RUNT:UPPer <UpperLevel>
Sets the upper voltage threshold for runt detection. A negative runt crosses the upper level twice without crossing the lower level.

Parameters:
<UpperLevel> Default unit:V

*RST: 600E-3

2.6.6 Rise / Fall time

SEARch:TRIGger:RISetime:TIME <Time>
Sets the reference rise or fall time, the nominal value for comparisons.

Parameters:
<Time>
Range: Depends on various settings, mainly time base and sample rate
Default unit: s

*RST: 200E-6

SEARch:TRIGger:RISetime:DELTa <DeltaTime>
Sets a range Δt to the reference rise/fall time set with `SEARch:TRIGger:RISetime:TIME`, if `SEARch:TRIGger:RISetime:RANGe` is set to Within or Outside. The instrument finds rise/fall times inside or outside the range time ± Δt.

Parameters:
<DeltaTime>
Range: Depends on various settings, mainly time base and sample rate
Default unit: s

*RST: 50e-6

SEARch:TRIGger:RISetime:SLOPe <Polarity>
Set the polarity for the Rise/fall time.

Parameters:
<Polarity>
POSitive | NEGative | EITHer

POSitive to search for rise time
NEGative to search for fall time
EITHer to search for rise and fall time

*RST: POS

SEARch:TRIGger:RISetime:RANGe <Range>
Sets how the measured rise or fall time is compared with the given limit(s).
To set the rise/fall time, use `SEARch:TRIGger:RISetime:TIME` .
To set the range ± Δt, use `SEARch:TRIGger:RISetime:DELTa` .

Parameters:

<Range> LONGer | SHORter | WITHin | OUTSide

LONGer
Finds rise/fall times longer than the given time.

SHORter
Finds rise/fall times shorter than the given time.

WITHin
Finds rise/fall times inside the range time ± Δt.

OUTSide
Finds rise/fall times outside the range time ± Δt.

*RST: LONG

SEARch:TRIGger:LEVel:RISetime:LOWer <LowerLevel>
Sets the lower voltage threshold. When the signal crosses this level, the rise time measurement starts or stops depending on the selected slope.

Parameters:

<LowerLevel> Default unit: V

*RST: 400E-3

SEARch:TRIGger:LEVel:RISetime:UPPer <UpperLevel>
Sets the upper voltage threshold. When the signal crosses this level, the rise/fall time measurement starts or stops depending on the selected slope.

Parameters:

<UpperLevel> Default unit: V

*RST: 600E-3

2.7 Quickmath, Mathematics and Reference Waveforms

2.7.1 Quickmath

This chapter describes commands that configure or perform basic math functions using Quickmath.

CALCulate:QMATh:STATe <State>
Defines whether the Quickmath waveform is active or not. Only if the source channels of the quick math waveform is available, it will be visible on the screen and can be selected as a source for analysis and display functions. Quickmath is only available in YT and Zoom mode.

Parameters:

<State> ON | OFF

*RST: OFF

CALCulate:QMATh:SOURce <m> <Source>
Defines the source of the Quickmath waveform.

Parameters:

<m> 1...2
Selects the source.

<Source> CH1 | CH2 | CH3 | CH4

Example:
```
CALC:QMAT:SOUR1 CH2
CALC:QMAT:SOUR2 CH3
```

*RST: 1

CALCulate:QMATh:OPERation <Operation>
Defines the operation of the Quickmath waveform.

Parameters:

<Operation> ADD | SUB | MUL | DIV

Example:
CALC:QMAT:OPER ADD in case of QMA=CH2 + CH3
CALC:QMAT:OPER SUB in case of QMA=CH2 - CH3
CALC:QMAT:OPER MUL in case of QMA=CH2 x CH3
CALC:QMAT:OPER DIV in case of QMA=CH2 / CH3

*RST: ADD

CALCulate:QMATh:SCALe <Scale>
Sets the vertical scale for the Quickmath waveform.

Parameters:

<Scale> Scale value, given in Volts per division.

Range: -1.0E-24 to 5.0E+25
Increment: 1, 2, 5 progression, for example, 1mV/div, 2mV/div, 5mV/div, 10, 20, 50...

*RST: 1

CALCulate:QMATh:POSition <Position>
Sets the vertical position of the Quickmath waveform in the window.

Parameters:

<Position> Position value, given in divisions
Increment: 0.01 in reset state

*RST: 0

CALCulate:QMATh:DATA?
Returns the data of the Quickmath waveform. The waveforms data can be used in MATHLAB, for example.
To set the export format, use „FORMat[:DATA]“ on page 50.

Return values:

<Data> List of values according to the format settings

Usage: Query only

CALCulate:QMATh:DATA:HEADer?
Returns information on the Quickmath waveform.

Table 2.4: Header data

Position	Meaning	Example
1	XStart in s	-9.477E-008 = - 94,77 ns
2	XStop in s	9.477E-008 = 94,77 ns
3	Record length of the waveform in Samples	200000
4	Number of values per sample interval, usually 1.	1

Return values:

<Header> Comma-separated value list

Example: -9.477E-008,9.477E-008,200000,1

Usage: Query only

CALCulate:QMATh:DATA:XINCrement?
Return the time difference between two adjacent samples of the indicated waveform.
The commands are relevant for data conversion if binary data format is defined (FORM UINT, 8|16).

Return values:

<Xincrement> Time in s

Usage: Query only

CALCulate:QMATh:DATA:XORigin?
Return time of the first sample depending to trigger event (time=0).
The commands are relevant for data conversion if binary data format is defined (FORM UINT, 8|16).

Return values:
<Xorigin> Time in s

Usage: Query only

CALCulate:QMATh:DATA:YINCrement?
Return the voltage value per bit of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Return values:
<Yincrement> Voltage in V

Usage: Query only

CALCulate:QMATh:DATA:YORigin?
Return the voltage value for binary value 0 of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Return values:
<Yorigin> Voltage in V

Usage: Query only

CALCulate:QMATh:DATA:YRESolution?
Return the vertical bit resolution of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Return values:
<Yresolution> For default waveforms, the resolution is 16 bit.

Usage: Query only

2.7.2 Mathematics

This chapter describes commands that configure or perform mathematical functions.

CALCulate:MATH<m>**:STATe** <State>
Defines whether the selected mathematical channel is active or not. Only if a channel is active it is visible on the screen and can be selected as a source for analysis and display functions.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<State> ON | OFF

*RST: OFF

CALCulate:MATH<m>**:SCALe** <Scale>
Sets the vertical scale for the specified math waveform.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<Scale> Scale value, given in Volts per division.

Range: -1.0E-24 to 5.0E+25
Increment: 1, 2, 5 progression, for example, 1mV/div, 2mV/div, 5mV/div, 10, 20, 50...

*RST: 1

CALCulate:MATH<m>**:POSition** <Position>
Sets the vertical position of the specified math waveform in the window.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<Position> Position value, given in divisions.
Increment: 0.01 in reset state

*RST: 0

CALCulate:MATH<m>**[:EXPRession][:DEFine]** <RemComplExpr>
Defines the equation to be calculated for the selected math channel as a regular expression.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<RemComplExpr> String with regular expression

Example:
Multiplication: `CALC:MATH1:EXPR:DEF "MUL(CH1,CH2)"`
Integral: `CALC:MATH2:EXPR:DEF "INT(MA1)"`
IIR low pass: `CALC:MATH3:EXPR:DEF "IIRL(MA2,1E6)"`

The mathematical expression is a string parameter, consisting of a keyword which represents the math function, followed by the respective sources.
These may be separated by comma and written in brackets.
A query of the math function always returns the string, which is listed under string expression.

Function	ExpressionString	Comment
Addition	"ADD(CH1,CH2)"	Alternativ "CH1+CH2"
Subtraction	"SUB(CH1,CH2)"	Alternativ "CH1-CH2"
Multiplication	"MUL(CH1,CH2)"	Alternativ "CH1*CH2"
Division	"DIV(CH1,CH2)"	Alternativ "CH1/CH2"
Max. amplitude	"MAX(CH1,CH2)"	
Min. amplitude	"MIN(CH1,CH2)"	
Square	"SQR(CH1)"	
Square root	"SQRT(CH1)"	
Absolute value	"ABS(CH1)"	
Positive wave	"POS(CH1)"	
Negative wave	"NEG(CH1)"	
Reciprocal 1/x	"REC(CH1)"	
Inverse	"INV(CH1)"	
Common logarithm	"LOG(CH1)"	
Natural logarithm	"LN(CH1)"	
Derivative	"DERI(CH1)"	
Integral	"INT(CH1)"	
IIR low pass	"IIRL(CH1,1E6)"	CH1 – Source 1e6 – Constant, limit frequency of the low pass
IIR high pass	"IIRH(CH1,1E6)"	CH1 – Source 1e6 – Constant, limit frequency of the high pass
FFT	"FFTMAG(CH1)"	FFT function of the source waveform

Table 2.5: Math expression

CALCulate:MATH<m>:DATA?
Returns the data of the math waveform. The waveforms data can be used in MATHLAB, for example.
To set the export format, use „FORMat[:DATA]“ on page 50.

Suffix:
<m> 1...4
Selects the math waveform.

Return values:
<Data> List of values depending on defined data format.

Usage: Query only

CALCulate:MATH<m>:DATA:HEADer?
Returns information on the math waveform.

Table 2.6: Header data

Position	Meaning	Example
1	XStart in s	-9.477E-008 = - 94,77 ns
2	XStop in s	9.477E-008 = 94,77 ns
3	Record length of the waveform in Samples	200000
4	Number of values per sample interval, usually 1.	1

Suffix:
<m> 1...4
Selects the math waveform.

Return values:
<Header> Comma-separated value list

Example: -9.477E-008,9.477E-008,200000,1

Usage: Query only

CALCulate:MATH<m>:LABel <Label>
Sets the label for the Math waveform.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<Label> String value
“xxxxxxxx“ (maximum 8 characters)

Example: „Power“

CH1=Voltage
CH2=Current
CH1*CH2=POWER

CALCulate:MATH<m>:LABel:STATe <State>
Switches the label of the math channel on or off.

Suffix:
<m> 1...4
Selects the math waveform.
Parameters:
<State> ON | OFF

*RST: OFF

CALCulate:MATH<m>:DATA:XINCrement?
Return the time difference between two adjacent samples of the indicated waveform.
The commands are relevant for data conversion if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the math waveform.

Return values:
<Xincrement> Time in s

Usage: Query only

CALCulate:MATH<m>:DATA:XORigin?
Return the time of the first sample of the indicated waveform.
The commands are relevant for data conversion if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the math waveform.

Return values:
<Xorigin> Time in s

Usage: Query only

CALCulate:MATH<m>:DATA:YINCrement?
Return the voltage value per bit of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the math waveform.

Return values:
<Yincrement> Voltage in V

Usage: Query only

CALCulate:MATH<m>:DATA:YORigin?
Return the voltage value for binary value 0 of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the math waveform.

Return values:
<Yorigin> Voltage in V

Usage: Query only

CALCulate:MATH<m>:DATA:YRESolution?
Return the vertical bit resolution of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1..4
Selects the math waveform.

Return values:
<Yresolution> For default waveforms, the resolution is 16 bit.

Usage: Query only

2.7.3 Reference Waveforms

REFCurve<m>:STATe
Displays or hides the selected reference waveform.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Parameters:
<State> ON | OFF

*RST: OFF

REFCurve<m>:SOURce <Source>
Defines the source of the reference waveform.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4 | POD1 | POD2 | QMA | MA1 | MA2 | MA3 | MA4 | MA5 | RE1 | RE2 | RE3 | RE4

CH1 | CH2 | CH3 | CH4
Signal channels 1 to 4

POD1 | POD2
Logic POD

QMA
Quick math channel

MA1 | MA2 | MA3 | MA4 | MA5
Math channels 1 to 5

RE1 | RE2 | RE3 | RE4
Reference channels 1 to 4

Any active channel, math, or reference waveform depend on the type of the instrument.

*RST: CH1

REFCurve<m>:SOURce:CATalog?
Returns a list of waveforms that can be used as reference source. Only available waveforms can be used.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Return values:
<Catalog> CH1 | CH2 | CH3 | CH4 | POD1 | POD2 | QMA | MA1 | MA2 | MA3 | MA4 | RE1 | RE2 | RE3 | RE4

CH1 | CH2 | CH3 | CH4
Signal channels 1 to 4

POD1 | POD2
Logic POD

QMA
Quick math channel

MA1 | MA2 | MA3 | MA4 | MA5
Math channels 1 to 5

RE1 | RE2 | RE3 | RE4
Reference channels 1 to 4

Any active channel, math, or reference waveform depend on the type of the instrument.

Usage: Query only

REFCurve<m>:UPDate
Updates the selected reference by the waveform defined with „REFCurve<m>:SOURce“ on page 99.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Usage: Event

REFCurve<m>:SAVE <FileName>
Stores the reference waveform the specified file.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Setting Parameters:
<FileName> String with path and file name

Usage: Setting only

REFCurve<m>**:LOAD** <FileName>
Loads the waveform data from the indicated reference file to the reference storage.
To load the instrument settings, use `REFCurve<m>:LOAD:STATe` (see command below):

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Setting Parameters:
<FileName> String with path and file name

Usage: Setting only

REFCurve<m>**:LOAD:STATe**
Loads the instrument settings in addition to the reference waveform data. The waveform data must be loaded before the settings, see `REFCurve<m>:LOAD` (see command above):
The settings are only available if the file was stored to the internal storage `/INT/REFERENCE` and never written to an external storage (USB stick).

Suffix:
<m> 1...4
Selects the reference waveform.

Usage: Event

REFCurve<m>**:HORizontal:SCALe** <Scale>
Changes the horizontal scale (timebase) of the reference waveform independent of the channel waveform settings.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Parameters:
<Scale> Default unit: s/div

*RST: 100e-6

REFCurve<m>**:HORizontal:POSition** <Position>
Changes the horizontal position of the reference waveform independent of the channel waveform settings.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Parameters:
<Position> Default unit: s

*RST: 0

REFCurve<m>**:VERTical:SCALe** <Scale>
Changes the vertical scale of the reference waveform.

Suffix:
<m> 1..4
Selects the reference waveform, the internal reference storage.

Parameters:
<Scale> Default unit: V/div

*RST: 1

REFCurve<m>:VERTical:POSition <Position>
Changes the vertical position of the reference waveform.

Suffix:
<m> 1..4
Selects the reference waveform, the internal reference storage.

Parameters:
<Position> Default unit: div
*RST: 0

REFCurve<m>:DATA?
Returns the data of the reference waveform for transmission from the instrument to the controlling computer. The waveforms data can be used in MATLAB, for example.

To set the export format, use „FORMat[:DATA]“ on page 50.

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Return values:
<Data> Comma-separated value list (depending on the defined data format)

Usage: Query only

REFCurve<m>:DATA:HEADer?
Returns information on the reference waveform.

Table 2.7: Header data

Position	Meaning	Example
1	XStart in s	-9.477E-008 = - 94,77 ns
2	XStop in s	9.477E-008 = 94,77 ns
3	Record length of the waveform in Samples	200000
4	Number of values per sample interval, usually 1.	1

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Parameters:
<Header> Comma-separated value list

Example: `-9.477E-008,9.477E-008,200000,1`

Usage: Query only

REFCurve<m>:DATA:XINCrement?
Return the time difference between two adjacent samples of the indicated waveform.
The commands are relevant for data conversion if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Return values:
<Xincrement> Time in s

Usage: Query only

REFCurve<m>:DATA:XORigin?
Return the time of the first sample of the indicated waveform.
The commands are relevant for data conversion if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Return values:
<Xorigin> Time in s

Usage: Query only

REFCurve<m>:DATA:YINCrement?
Return the voltage value per bit of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Return values:
<Yincrement> Voltage in V

Usage: Query only

REFCurve<m>:DATA:YORigin?
Return the voltage value for binary value 0 of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Return values:
<Yorigin> Voltage in V

Usage: Query only

REFCurve<m>:DATA:YRESolution?
Return the vertical bit resolution of the indicated waveform.
The commands are relevant for data conversion, if binary data format is defined (FORM UINT, 8|16).

Suffix:
<m> 1...4
Selects the reference waveform, the internal reference storage.

Return values:
<Yresolution> For default waveforms, the resolution is 8 bit.
Its resolution is depending on the source waveform. The maximum is 16 bit.

Usage: Query only

2.8 FFT

CALCulate:MATH<m>:ARIThmetics <Arithmetics>
Defines the mode for FFT calculation and display.

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Parameters:
<Arithmetics> OFF | ENVelope | AVERage

OFF
The FFT is performed without any additional weighting or postprocessing of the acquired data. The new input data is acquired and displayed, and thus overwrites the previously saved and displayed data.

ENVelope
In addition to the normal spectrum, the maximal oscillations are saved separately and updated for each new spectrum. The maximum values are displayed together with the newly acquired values and form an envelope. This envelope indicates the range of all FFT trace values that occurred.

AVERage
The average of several spectrums is calculated. The number of spectrums used for the averaging is defined using the command `CALCulate:MATH<m>:ARIThmetics`.
This mode is useful for noise rejection.

*RST: OFF

CALCulate:MATH<m>:FFT:AVERage:COUNt
Defines the number of spectrums used for averaging if `CALCulate:MATH<m>:ARIThmetics` (see command above) is set to AVERage.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<AverageCount> Integer value
Range: 2 to 512
Increment: 2^n

*RST: 2

CALCulate:MATH<m>:FFT:MAGNitude:SCALe
Defines the scaling of the y-axis.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<Magnitude Scale> LINear | DBM | DBV

LINear: linear scaling; displays the RMS value of the voltage.
DBM: logarithmic scaling; related to 1 mW
DBV: logarithmic scaling; related to 1 V_{eff}

*RST: DBM

CALCulate:MATH<m>:FFT:BANDwidth[:RESolution]:ADJusted?
Queries the effective resolution bandwidth.

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Return values:
<AdjResBW> Range: -100e24 to 100e24
Increment: 0.1
*RST: 1.221E3
Default unit: Hz

Usage: Query only

CALCulate:MATH<m>:FFT:BANDwidth[:RESolution]:AUTO <SpanRBWCoupling>
Couples the frequency span to the RBW.

Suffix:
<m> 1..4
The numeric suffix is irrelevant.

Parameters:
<SpanRBWCoupling> ON | OFF

CALCulate:MATH<m>:FFT:BANDwidth[:RESolution]:RATio <SpanRBWRatio>
Defines the ratio resolution bandwidth (Hz) / span (Hz).

Suffix:
<m> 1..4
The numeric suffix is irrelevant.

Parameters:
<SpanRBWRatio> Range: 1024 to 32768
Increment: 2^n

*RST: 4096

CALCulate:MATH<m>:FFT:BANDwidth[:RESolution][:VALue] <ResolutionBW>
Defines the resolution bandwidth.

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Parameters:
<ResolutionBW> Range:Depends on various other settings.
Default unit:Hz

CALCulate:MATH<m>:FFT:CFRequency <CenterFreq>
Defines the position of the displayed frequency domain, which is (Center - Span/2) to (Center + Span/2). The width of the domain is defined using the `CALCulate:MATH<m>:FFT:SPAN` command.

Suffix:
<m> 1...4
Selects the math waveform.

Parameters:
<CenterFreq> Range: 0 to 2e12
Increment: 1
Default unit: Hz

*RST: 500e6

CALCulate:MATH<m>:FFT:FULLspan
Performs FFT calculation for the full frequency span.

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Usage: Event

CALCulate:MATH<m>:FFT:SPAN <FreqSpan>
The span is specified in Hertz and defines the width of the displayed frequency range, which is (Center - Span/2) to (Center + Span/2). The position of the span is defined using the `CALCulate:MATH<m>:FFT:CFRequency` command.

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Parameters:
<FreqSpan> Default unit: Hz

CALCulate:MATH<m>:FFT:STARt <StartFreq>
Defines the start frequency of the displayed frequency domain (instead of defining a center frequency and span).

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Parameters:
<StartFreq> Default unit: Hz

CALCulate:MATH<m>**:FFT:STOP** <StopFreq>
Defines the stop frequency of the displayed frequency domain (instead of defining a center frequency and span).

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Parameters:
<StopFreq> Default unit: Hz

CALCulate:MATH<m>**:FFT:WINDow:TYPE** <WindowType>
Window functions are multiplied with the input values and thus can improve the FFT display.

Suffix:
<m> 1...4
The numeric suffix is irrelevant.

Parameters:
<WindowType> RECTangular | HAMMing | HANNing | BLACkmanharris

RECTangular
The rectangular window multiplies all points by one. The result is a high frequency accuracy with thin spectral lines, but also with increased noise. Use this function preferably with pulse response tests where start and end values are zero.

HAMMing
The Hamming window is bell shaped. Its value is not zero at the borders of the measuring interval. Thus, the noise level inside the spectrum is higher than Hanning or Blackman, but smaller than the rectangular window. The width of the spectral lines is thinner than the other bell-shaped functions. Use this window to measure amplitudes of a periodical signal precisely.

HANNing
The Hanning window is bell shaped. Unlike the Hamming window, its value is zero at the borders of the measuring interval. Thus, the noise level within the spectrum is reduced and the width of the spectral lines enlarges. Use this window to measure amplitudes of a periodical signal precisely.

BLACkmanharris
The Blackman window is bell shaped and has the steepest fall in its wave shape of all other available functions. Its value is zero at both borders of the measuring interval. In the Blackman window the amplitudes can be measured very precisely. However, determining the frequency is more difficult. Use this window to measure amplitudes of a periodical signal precisely.

*RST: HANNing

CALCulate:MATH<m>**:FFT:TIME:RANGe** <WindowWidth>
Defines the width of the time base extract from the Y(t)-window for which the FFT is calculated.

Parameters:
<WindowWidth> Range: depends on the time base
Default unit: s

CALCulate:MATH<**m**>**:FFT:TIME:POSition** <WindowPosition>
Defines the position of the time base extract in the Y(t)-window for which the FFT is calculated.

Parameters:

<WindowPosition>	Range:	depends on the time base and the width of the FFT time base extract
	Default unit:	s

CALCulate:MATH<**m**>**:FFT:SRATe?** <SampleRate>
Returns the sample rate of data used in an FFT analysis.

Return values:

<SampleRate>	Default unit:	Sa/s

Usage: Query only

2.9 Masks

MASK:STATe <State>
Turns the mask test mode on or off. When turning off, any temporarily stored new masks are deleted.

Parameters:
<State> ON | OFF

*RST: OFF

MASK:TEST <Test>
Starts, finishes or interrupts a mask test.

Parameters:
<Test> RUN | STOP

*RST: STOP

MASK:LOAD <FileName>
Loads a stored mask from the specified file.

Setting Parameters:
<FileName> String parameter
Path and file name

Usage: Setting only

MASK:SAVE <FileName>
Saves the current mask in the specified file.

Setting Parameters:
<FileName> String parameter
Path and file name

Usage: Setting only

MASK:SOURce <Source>
Defines the channel to be compared with the mask.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4

*RST: CH1

MASK:CHCopy
Creates a mask from the envelope waveform of the test source set with `MASK:SOURce`.

Usage: Event

MASK:YPOSition <Yposition>
Moves the mask vertically within the display.

Parameters:
<Yposition> Mask offset from the vertical center
Range: -200 to 200
Increment: 0.02
Default unit: div

*RST: 0

MASK:YSCALe <Yscale>
Changes the vertical scaling to stretch or compress the mask in y-direction.

Parameters:
<Yscale> A value over 100% stretches the amplitudes; a value less than 100% compresses the amplitudes.
Range: 10 to 1000
Increment: 1
Default unit: %

*RST: 100

MASK:YWIDth <Yaddition>
Changes the width of the mask in vertical direction.

Parameters:
<Yaddition> The value is added to the y-values of the upper mask limit and subtracted from the y-values of the lower mask limit.
Range: 0 to 5.12
Increment: 0.04
Default unit: div

*RST: 0

MASK:XWIDth <Xaddition>
Changes the width of the mask in horizontal direction.

Parameters:
<Xaddition> The value is added to the positive x-values and subtracted from the negative x-values of the mask limits in relation to the mask center.

Range: 0 to 10
Increment: 0.01
Default unit: div

*RST: 0

MASK:COUNt?
Returns the number of tested acquisitions.

Return values:
<TotalCount> Total number of tested acquisitions

Usage: Query only

MASK:VCOunt?
Returns the number of acquistions that hit the mask.

Return values:
<ViolationCount> Number of violated acquisitions

Usage: Query only

MASK:ACTion:YOUT:ENABLe <Yout>
Enable or disable of pulses at the Y-OUTput if mask is distorted.

Parameters:
<Yout> ON | OFF

*RST: OFF

2.10 Component Tester (HMO72x....202x only)

COMPonenttest:STATe <State>
Switches the component tester mode on or off.

Parameters:
<State> ON | OFF

*RST: OFF

COMPonenttest:FREQuency <Frequency>
Sets the frequency of the component tester source.

Parameters:
<Frequency> F50 | F200

F50
Sets the frequency of the component tester source to 50Hz.

F200
The frequency is 200 Hz.

2.11 Protocol Analysis

2.11.1 General

BUS<b>:STATe <State>
Switches the protocol display on or off.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<State> ON | OFF

*RST: OFF

BUS<b>:TYPE <Type>
Defines the bus or interface type for analysis. For most types, a special option to the instrument is required.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Type> PARallel | CPARallel | I2C | SPI | SSPI | UART | CAN | LIN

*RST: PARallel

BUS<b>:FORMat <Format>
Sets the decoding format for the display on the screen.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Format> ASCii | HEXadecimal | BINary | DECimal

*RST: HEXadecimal

BUS<**b**>**:DSIZe** <DisplaySize>
Sets the height of the decoded bus signal on the screen.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<DisplaySize> SMALl | MEDium | LARGe

*RST: MEDium

BUS<**b**>**:DSIGnals** <BitsSignals>
Displays the individual bit lines above the decoded bus line.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BitsSignals> ON | OFF

*RST: ON

BUS<**b**>**:POSition** <Position>
Sets the vertical position of the decoded bus signal in divisions on the screen.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Position> Increment: 0.02
Default unit: div

*RST: -3.5

BUS<**b**>**:LABel** <Label>
Sets the label for the bus.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Label> String value
„UART_PC" (maximum 8 characters)

BUS<**b**>**:LABel:STATe** <State>
Switches the label of the bus on or off.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<State> ON | OFF

*RST: ON

SOURce:FUNCtion[:SHAPe] <SignalShape>
Define the function of the internal bus signal source.

Parameters:
<SignalShape> SQUare | RANDom | I2C | SPI | COUNt | UART

*RST: SQUare

SOURce:FREQuency <Frequency>
Define the frequency of the internal bus signal source.

Parameters:
<Frequency> numeric value, depending of the function choosen for the internal bus signal source.
Square, Random: 1000, 1000000
UART: 9600, 115200, 1000000
SPI: 100000, 250000, 1000000
I^2C: 100000, 400000, 1000000
Count: 1000, 1000000

*RST: 1000

2.11.2 Parallel Bus

BUS<b>:PARallel:DATA<m>:SOURce <Source>
Sets the input channels for the data bit lines. The suffix <m> selects the bit line.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Source> D0 .... D15

*RST: D0

BUS<b>:PARallel:WIDTh <BusWidth>
Sets the number of lines to be analyzed.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BusWidth> Maximum number depends on HMO model.
Range: 1 to 16
Increment: 1
Default unit: Bit

*RST: 4

2.11.3 SPI

SPI - Configuration

Trigger

Decode

BUS<b>:SPI:CS:SOURce <Source>
Selects the input channel of the chip select line.

Suffix:

<b> 1, 2
Select the bus.

Parameters:

<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15 | EXTern

The external trigger input of the 2-channel instruments (HMO722, HMO1022, HMO1522, HMO2022, HMO3522) can be used as chip select source.

*RST: CH1

BUS<b>:SPI:CS:POLarity <Polarity>
Selects whether the chip select signal is high active (high = 1) or low active (low = 1).

Suffix:

<b> 1, 2
Select the bus.

Parameters:

<Polarity> POSitive | NEGative

POSitive = high active
NEGative = low active

*RST: POSitive

BUS<b>:SPI:CLOCk:SOURce <Source>
Selects the input channel of the clock line.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:SPI:CLOCk:POLarity <Polarity>
Selects if data is stored with the rising or falling slope of the clock. The slope marks the begin of a new bit.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Polarity> POSitive | NEGative

POSitive: rising slope
NEGative: falling slope

*RST: NEGative

BUS<b>:SPI:DATA:SOURce <Source>
Selects the input channel of the data line.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:SPI:DATA:POLarity <Polarity>
Selects whether transmitted data is high active (high = 1) or low active (low = 1).

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Polarity> POSitive | NEGative

POSitive = high active
NEGative = low active

*RST: POSitive

BUS<b>:SPI:BORDer <BitOrder>
Defines if the data of the messages starts with MSB (most significant bit) or LSB (least significant bit).

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BitOrder> MSBFirst | LSBFirst

*RST: MSBFirst

BUS<b>:SPI:SSIZe <SymbolSize>
Sets the word length, the number of bits in a message.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<SymbolSize> Range: 4 to 32
Increment: 1
Default unit: Bit

*RST: 8 Bit

TRIGger:A:SPI:MODE <Mode>
Specifies the trigger mode for SPI/SSPI protocols.

Parameters:
<Mode> BSTart | BEND | NTHBit | PATTern

BSTart
Burst start, sets the trigger event to the start of the frame. The frame starts when the chip select signal CS changes to the active state. In case of SSPI protocol the frames starts with the first slope of the clock signal.

BEND
Burst end, sets the trigger event to the end of the message. This event occurs if the CS signal changes to inactive. In case of an SSPI protocol the frames ends if the idle time has expired after the last clock slope.

NTHBit
Sets the trigger event to the specified bit number. To define the bit number, use `TRIGger:A:SPI:POFFset`.

PATTern
Sets the trigger event to a serial pattern. To define the pattern, use `TRIGger:A:SPI:PATTern`.
For a complete configuration of the pattern mode, you also have to set `TRIGger:A:SPI:PLENgth` and `TRIGger:A:SPI:POFFset`.

*RST: BSTart

TRIGger:A:SPI:PATTern <DataPattern>
Defines the bit pattern as trigger condition.

Parameters:
<DataPattern> Binary pattern with max. 32 bit. Characters 0, 1, and X are allowed.

Example: `TRIGger:A:SPI:PATTern "0011XXXX0110"`
Sets a 12bit pattern.

TRIGger:A:SPI:PLENgth <PatternLength>
Defines how many bits build up the serial pattern.

Parameters:
<PatternLength> Range: 1 to 32
Increment: 1

*RST: 4

TRIGger:A:SPI:POFFset <PatternBitOffset>
Sets the number of bits before the first bit of the pattern.

Parameters:
<PatternBitOffset> Number of ignored bits
Range: 0 to 4095
Increment: 1

*RST: 0

BUS<**b**>**:SPI:FCOunt?**<FrameCount>
Returns the number of frames.

Suffix:
<b> 1..2
Select the bus.

Return values:
<FrameCount> Total number of decoded frames.

Usage: Query only

BUS<b>:SPI:FRAMe<n>:STATus? <Status>
Returns the status of frames.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame.

Return values:
<Status> OK | INCFirst | INCLast | INSufficient

OK	frame is o.k.
INCFirst	first frame is incomplete
INCLast	last frame is incomplete
INSufficient	frame is insufficient

Usage: Query only

BUS<b>:SPI:FRAMe<n>:STARt? <StartTime>
Returns the start time of a frame.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame.

Return value:
<StartTime>

Range:	depends on sample rate, record length, and time base
Increment:	depends on the time base
Default unit:	s

Usage: Query only

BUS<b>:SPI:FRAMe<n>:STOP? <StopTime>
Returns the stop time of the frame.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame.

Return value:
<StopTime>

Range:	depends on sample rate, record length, and time base
Increment:	depends on the time base
Default unit:	s

Usage: Query only

BUS<b>:SPI:FRAMe<n>:DATA? <Data>
Returns the comma-separeted frame data.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame.

Return values:
<Data> List of decimal values of data bytes

Example: BUS:SPI:FRAM3:DATA?

BUS<b>:SPI:FRAMe<n>:WCOunt? <WordCount>
Returns the number of words of a frame.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame.

Return value:
<WordCount> Number of words

Usage: Query only

BUS<b>:SPI:FRAMe<n>:WORD<o>STARt? <StartTime>
Gives back the start time of a word of a frame.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame.

<o> *
Selects the word number.

Return value:
<StartTime> Range: depends on sample rate, record length, and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<b>:SPI:FRAMe<n>:WORD<o>STOP? <StopTime>
Returns the stop time of a word of a frame.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame.

<o> *
Selects the word number.

Return value:
<StopTime> Range: depends on sample rate, record length, and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<b>:SPI:FRAMe<n>:WORD<o>MISO? <Data>
Returns the decimal value of a data word of a frame.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame (1...n).

<o> *
Selects the word number (1...o).

Return values:
<Data> Decimal value of a data word

Usage: Query only

BUS<b>:SPI:FRAMe<n>:WORD<o>MOSI? <Data>
Returns the decimal value of a data word of a frame.

Suffix:
<b> 1..2
Select the bus.

<n> *
Selects the frame (1...n).

<o> *
Selects the word number (1...o).

Return values:
<Data> Decimal value of a data word

Usage: Query only

2.11.4 SSPI

BUS<b>:SSPI:CLOCk:SOURce <Source>
Selects the input channel of the clock line.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:SSPI:CLOCk:POLarity <Polarity>
Selects if data is stored with the rising or falling slope of the clock. The slope marks the begin of a new bit.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Polarity> POSitive | NEGative

POSitive: rising slope
NEGative: falling slope

*RST: POSitive

BUS<b>:SSPI:DATA:SOURce <Source>
Selects the input channel of the data line.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:SSPI:DATA:POLarity <Polarity>
Selects whether transmitted data is high active (high = **1**) or low active (low = **1**).

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Polarity> POSitive | NEGative

POSitive = high active
NEGative = low active

*RST: POSitive

BUS<b>:SSPI:BITime <BurstIdleTime>
Within the idle time the data and clock lines are low. A new frame begins when the idle time has expired and the clock line has been inactive during that time. If the time interval between the data words is shorter than the idle time, the words are part of the same frame.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BurstIdleTime> Range: 16e-9 to 1E-3
Increment: 16e-9
Default unit: s

*RST: 100e-6

BUS<b>:SSPI:BORDer <BitOrder>
Defines if the data of the messages starts with MSB (most significant bit) or LSB (least significant bit).

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BitOrder> MSBFirst | LSBFirst

*RST: MSBFirst

BUS<b>:SSPI:SSIZe <SymbolSize>
Sets the word length, the number of bits in a message.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<SymbolSize> Range: 4 to 32
Increment: 1
Default unit: Bit

*RST: 8 Bit

2.11.5 I²C

I²C - Configuration

Trigger

Decode

BUS<b>:I2C:CLOCk:SOURce <Source>
Sets the input channel to which the clock line is connected.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:I2C:DATA:SOURce <Source>
Sets the input channel to which the data line is connected.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

TRIGger:A:I2C:MODE <**Mode**>
Specifies the trigger mode for I^2C.

Parameters:
<Mode> STARt | RESTart | STOP | MACKnowledge | PATTern

STARt
Start of the message. The start condition is a falling slope on SDA while SCL is high.

RESTart
Restarted message. The restart is a repeated start condition.

STOP
End of the message. The stop condition is a rising slope on SDA while SCL is high.

MACKnowledge
Missing acknowledge. If the transfer failed, at the moment of the acknowledge bit the SCL and the SDA lines are both on high level.

PATTern
Triggers on a set of trigger conditions:read or write access of the master, to an address, or/and to a bit pattern in the message.
For a complete configuration of the pattern mode, you have to set:
`TRIGger:A:I2C:ACCess` (read/write access), and
`TRIGger:A:I2C:AMODe` and `TRIGger:A:I2C:ADDRess` (address), and/or
`TRIGger:A:I2C:POFFset` and `TRIGger:A:I2C:PLENgth` and
`TRIGger:A:I2C:PATTern` (pattern)

*RST: STARt

TRIGger:A:I2C:ACCess <Access>
Toggles the trigger condition between Read and Write access of the master.

Parameters:
<Access> READ | WRITe

*RST: READ

TRIGger:A:I2C:AMODe <AdrMode>
Sets the lenght of the slave address.

Parameters:
<AdrMode> NORMal | EXTended

NORMal: 7 bit address
EXTended: 10 bit address

*RST: NORMal

TRIGger:A:I2C:ADDRess <AddressString>
Sets the address of the slave device. The address can have 7 bits or 10 bits.

Parameters:
<AddressString> Binary pattern with max. 10 bit. Characters 0, 1, and X are allowed.

Example: `TRIG:A:I2C:ADDR "10x1"`

TRIGger:A:I2C:PATTern <DataPattern>
Defines the bit pattern as trigger condition. Make sure that the correct pattern length has been defined before with `TRIGger:A:I2C:PLENgth` (see command below).

Parameters:

<DataPattern> String with max. 24 characters. Characters 0, 1, and X are allowed. X can be assigned to a specified bit. If you define a pattern shorter than the pattern length, the missing LSB are filled with X. If you define a pattern longer than the pattern length, the pattern string is not valid.

Example:

```
TRIG:A:I2C:PLEN 2
TRIG:A:I2C:PATT „10X10000XXXX1111"
TRIG:A:I2C:PATT?
```
Return value (2 bytes):„10X10000XXXX1111"

Example:

```
TRIG:A:I2C:PLEN 1
TRIG:A:I2C:PATT „110"
TRIG:A:I2C:PATT?
```
Return value (1 byte):„110XXXXX"

TRIGger:A:I2C:PLENgth <PatternLength>
Defines how many bytes are considered in the trigger condition. To set the pattern for these bytes, use `TRIGger:A:I2C:PATTern` (see command above).

Parameters:

<PatternLength> Number of bytes
Range: 1 to 3
Increment: 1

*RST: 1

TRIGger:A:I2C:POFFset <PatternByteOffset>
Sets the number of bytes before the first byte of interest, relating to the end of the address bytes.

Parameters:

<PatternByteOffset> Number of ignored bytes
Range: 0 to 4095
Increment: 1

*RST: 0

BUS<b>:I2C:FCOunt? <FrameCount>
Returns the number of frames.

Suffix:

<b> 1, 2
Select the bus.

Return values:

<FrameCount> Total number of decoded frames.

Usage: Query only

BUS<b>:I2C:FRAMe<n>:STATus? <State>
Returns the status of the selected frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<State> INComplete | OK | UNEXpstop | INSufficient | ADDifferent

INComplete frame is incomplete
OK frame is o.k.
UNEXpstop frame stop is unexpected
INSufficient frame is insufficient
ADDifferent frame address is different

Usage: Query only

BUS<b>:I2C:FRAMe<n>:STARt? <StartTime>
Returns the start time of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<StartTime> Range: depends on sample rate, record length, and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<b>:I2C:FRAMe<n>:STOP? <EndTime>
Returns the stop time of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<EndTime> Range: depends on sample rate, record length, and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<b>:I2C:FRAMe<n>:ACCess? <Access>
Returns the transfer direction - read or write access from master to slave.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<Access> INComplete | READ | WRITE | EITHer | UNDF

INComplete	frame is incomplete
READ	read frame
WRITE	write frame
EITHer	either write or read frame
UNDF	frame is undefined

Usage: Query only

BUS<b>:I2C:FRAMe<n>:AMODe? <AddressMode>
Returns the address length.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<AddressMode> BIT7 | BIT10

BIT7	7 Bit address
BIT10	10 Bit address

Usage: Query only

BUS<b>:I2C:FRAMe<n>:AACCess? <Acknowledge>
Returns the address acknowledge bit value for the indicated frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<Acknowledge> INComplete | ACK | NACK | EITHer

INComplete	acknowledge is incomplete
ACK	acknowledge
NACK	not acknowledge
EITHer	ack or nack

Usage: Query only

BUS<b>:I2C:FRAMe<n>:ADDRess? <AddressValue>
Returns the decimal address value of the indicated frame including the R/W bit.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<AddressValue> Decimal value
Range: 0 to 2047
Increment: 1

Usage: Query only

BUS<b>:I2C:FRAMe<n>:ADEVice? <SlaveAddress>
Returns the read/write address as a decimal value (excluding RW bit).

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<SlaveAddress> Decimal value
Range: 0 to 1023
Increment: 1

Usage: Query only

BUS<b>:I2C:FRAMe<n>:ASTart? <StartTime>
Returns the start time of the address for the indicated frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<StartTime> Range: depends on sample rate and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:ADBStart?** <AckStartTime>
Returns the start time of a address acknowledge of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<AckStartTime> Range: depends on sample rate and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:ACOMplete?** <AddressComplete>
Returns the state of the address.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<AddressComplete> ON | OFF

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:BCOunt?** <ByteCountInFrame>
Returns the number of data bytes in the specified frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<ByteCountInFrame> Number of words (bytes)

Example:
```
BUS:I2C:FRAM2:BCO?
-> 4
```

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:DATA?** <DataWordsInFrame>
Returns a list of data words of the specified frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

Return values:
<DataWordsInFrame> Comma-separated list of decimal values of the data bytes

Example: `BUS:I2C:FRAM2:DATA?`
returns four data bytes:
`-> 69,158,174,161`

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:BYTE**<**o**>**VALue** <ByteValue>
Returns the decimal value of the specified byte.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

<o> 1...o
Selects the byte number.

Return values:
<ByteValue> Decimal value
Range: 0 to 255
Increment: 1

Example: `BUS:I2C:FRAM2:BYTE2:VAL?`
`-> 158`

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:BYTE**<**o**>**STARt** <StartTime>
Returns the start time of the specified data byte.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

<o> *
Selects the byte number.

Return values:
<StartTime> Range: depends on sample rate and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:BYTE**<**o**>**:ACKStart** <AckStartTime>
Returns the start time of the acknowledge bit of a data byte of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

<o> *
Selects the byte number.

Return values:
<AckStartTime> Range: depends on sample rate and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:BYTE**<**o**>**:ACCess** <Acknowledge>
Returns the acknowledge bit of a data byte of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

<o> *
Selects the byte number.

Return values:
<Acknowledge> INComplete | ACK | NACK | EITHer

INComplete acknowledge is incomplete
ACK acknowledge
NACK not acknowledge
EITHer ack or nack

Usage: Query only

BUS<**b**>**:I2C:FRAMe**<**n**>**:BYTE**<**o**>**:COMPlete** <ByteComplete>
Returns the information whether a data byte of a frame is received complete.

Suffix:
<b> 1, 2
Select the bus.

<n> 1...n
Selects the frame.

<o> *
Selects the byte number.

Return values:
<ByteComplete> ON | OFF

Usage: Query only

2.11.6 UART

UART - Configuration

Trigger

Decode

BUS<b>:UART:DATA:SOURce <Source>
Selects the input channel of the UART signal.

Suffix:

<b> 1, 2
Select the bus.

Parameters:

<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:UART:DATA:POLarity <Polarity>
Defines if the transmitted data on the bus is high (high = 1) or low (low = 1) active.

Suffix:

<b> 1, 2
Select the bus.

Parameters:

<Polarity> POSitive | NEGative

POSitive = high active
NEGative = low active

*RST: POSitive

BUS<b>:UART:SSIZe <SymbolSize>
Sets the number of data bits in a message.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<SymbolSize> Range: 5 to 9
Increment: 1
Default unit: Bit

*RST: 8

BUS<b>:UART:PARity <Parity>
Defines the optional parity bit that is used for error detection.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Parity> ODD | EVEN | NONE

*RST: NONE

BUS<b>:UART:SBIT <StopBitNumber>
Sets the stop bits.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<StopBitNumber> B1 | B1_5 | B2
1; 1.5 or 2 stop bits are possible.

*RST: B1

BUS<b>:UART:BAUDrate <Baudrate>
Sets the number of transmitted bits per second.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Baudrate> Range: 100 to 78.1E6 (Upper limit depends on HMO model)
Default unit: Bit

*RST: 115200

BUS<b>:UART:BITime <BurstIdleTime>
Sets the minimal time between two data frames (packets), that is, between the last stop bit and the start bit of the next frame.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BurstIdleTime> Range: 12.8E-9 to 53.68E-3
Increment: 12.8E-9
Default unit: s

*RST: 64E-9

TRIGger:A:UART:MODE <Mode>
Specifies the trigger mode for UART/RS-232 interfaces.

Parameters:
<Mode> BSTart | SBIT | NTHSymbol | SYMBol | PATTern | PERRor | FERRor | BREak

BSTart
Burst start. Sets the trigger to the begin of a data frame. The frame start is the first start bit after the idle time.

SBIT
Start bit. The start bit is the first low bit after a stop bit.

NTHSymbol
Sets the trigger to the n-th symbol of a burst.

SYMBol
Triggers if a pattern occurs in a symbol at any position in a burst.

PATTern
Triggers on a serial pattern at a defined position in the burst. To define the pattern, use `TRIGger:A:UART:PLENgth` and `TRIGger:A:UART:PATTern`.
To define the position, use `TRIGger:A:UART:POFFset`.

PERRor
Parity Error:Triggers if a bit error occured in transmission.

FERRor
Triggers on frame error.

BREak
Triggers if a start bit is not followed by a stop bit within a defined time. During the break the stop bits are at low state.

*RST: SBIT

TRIGger:A:UART:PATTern <DataPattern>
Defines the bit pattern as trigger condition.

Parameters:
<DataPattern> Binary pattern with max. 32 bit. Characters 0, 1, and X are allowed.

*RST: 1 = „00000001"

TRIGger:A:UART:PLENgth <PatternLength>
Defines how many symbols build up the serial pattern.

Parameters:
<PatternLength> Number of symbols
Range: 1 to 6
The maximum number of symbols are depending on the symbol size of the UART protocol.

Symbol size	Max. symbols
5	6
6	5
7	4
8	4
9	3

Table 2.8: Symbols

Increment: 1
*RST: 1

TRIGger:A:UART:POFFset <PatternByteOffset>
Sets the number of symbols before the first symbol of the pattern.

Parameters:
<PatternByteOffset> Number of ignored symbols

Range: 0 to 4095
Increment: 1
*RST: 0

BUS<b>:UART:FCOunt? <FrameCount>
Returns the number of decoded frames on the on the data line.

Suffix
<b> 1, 2
Select the bus.
Return values:
<FrameCount> Total number of decoded frames.

Usage: Query only

BUS<b>:UART:FRAMe<n>:WORD<o>:VALue? <Value>
Return the value of the specified symbol (word) on the data line.

Response:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

<o> *
Selects the word number.
Return values:
<Value> Decimal value
Range: 0 to 511
Increment: 1

Usage: Query only

BUS<b>:UART:FRAMe<n>:WCOunt? <WordCount>
Returns the number of symbols in the specified frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<WordCount> Number of words (symbols, characters)

Usage: Query only

BUS<b>:UART:FRAMe<n>:WORD:STARt? <StartTime>
Returns the start time of the specified symbol (word).

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<StartTime> Range: depends on sample rate, record length and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<b>:UART:FRAMe<n>:WORD<o>:STOP? <StopTime>
Returns the stop time of a word of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<StopTime> Range: depends on sample rate, record length and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<**b**>**:UART:FRAMe**<**n**>**:WORD**<**o**>**:STATe?** <FrameStatus>
Returns the status of frames.

Suffix:

<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:

<FrameStatus> OK | FRSTart | FRENd | FRMError | STERror | SPERror | PRERror | INSufficient

OK	frame is o.k.
FRSTart	frame start
FRENd	frame end
FRMError	frame error
STERror	startbit error
SPERror	stopbit error
PRERror	parity error
INSufficient	frame is insufficient

*RST: OK

Usage: Query only

2.11.7 CAN

CAN - Configuration

Trigger

Decode

BUS<b>:CAN:DATA:SOURce <Source>
Sets the input channel to which the data line is connected.

Suffix:

<b> 1, 2
Select the bus.

Parameters:

<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:CAN:TYPE <SignalType>
Sets the signal type of the CAN.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<SignalType> CANH | CANL
CANH CAN High
CANL CAN Low

*RST: CANH

BUS<b>:CAN:SAMPlepoint <SamplePoint>
Sets the sample point for the CAN Decoder within the bit period.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<SamplePoint> <numeric_value>
Min.: 25
Max.: 90
Inc.: 1
Unit:%

*RST: 50

BUS<b>:CAN:BITRate <BitRate>
Sets the number of transmitted bits per second.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BitRate> <numeric_value>
HMO35xx: 100Bit/s to 4.03MBit/s
HMO2524: 100Bit/s to 2.52MBit/s
HMO72x...202x: 100Bit/s to 2.016MBit/s
Unit: Bit/s

*RST: 50E03

TRIGger:A:CAN:TYPE <TriggerType>
Specifies the trigger type for CAN.

Parameters:
<TriggerType> STOFrame | EOFrame | ID | IDDT | FTYPe | ERRCondition

STOFrame Sets the trigger to the start of a frame.
EOFrame Sets the trigger to the end of a frame.
ID Sets the trigger to the ID of a frame.
IDDT Sets the trigger to the ID and data of a frame.
FTYPe Sets the trigger to the frame type.
ERRCondition Sets the trigger to the error condition of a frame.

*RST: STOF

TRIGger:A:CAN:FTYPe <FrameType>
Specifies the frame type for CAN.

Parameters:

<FrameType>	DATA | REMote | ERRor | OVERload | ANY	
	DATA	Sets the frame type to data.
	REMote	Sets the frame type to remote.
	ERRor	Sets the frame type to error.
	OVERload	Sets the frame type to overload.
	ANY	Sets the frame type to any.
	*RST:	ERR

TRIGger:A:CAN:ITYPe <IdentifierType>
Specifies the identifier type for CAN.

Parameters:

<IdentifierType>	B11 | B29 | ANY	
	B11	Sets the identifier type 11Bit.
	B29	Sets the identifier type 29Bit.
	ANY	Sets the identifier type to any.
	*RST:	B11
	(if trigger type ID is set, only B11 or B29 is allowed)	

TRIGger:A:CAN:ICONdition <IdentifierCondition>
Specifies the condition for the identifier.

Parameters:

<IdentifierCondition>	EQUual | NEQual | GTHan | LTHan	
	EQUual	Sets the condition for identifier to equal.
	NEQual	Sets the condition for identifier to not equal.
	GTHan	Sets the condition for identifier to greater then.
	LTHan	Sets the condition for identifier to less then.
	*RST:	EQU

TRIGger:A:CAN:IDENtifier <Identifier>
Specifies the identifier, depending of `TRIGger:A:CAN:ITYPe` <IdentifierType> setting only string with 11 or 29 characters is possible.

Parameters:

<Identifier>	01X-string with up to 29 character

TRIGger:A:CAN:DCONdition <DataCondition>
Specifies the condition for the data.

Parameters:

<DataCondition>	EQUual | NEQual | GTHan | LTHan	
	EQUual	Sets the condition for identifier to equal.
	NEQual	Sets the condition for identifier to not equal.
	GTHan	Sets the condition for identifier to greater then.
	LTHan	Sets the condition for identifier to less then.
	*RST:	EQU

TRIGger:A:CAN:DATA <Data>
Specifies the data for CAN trigger. Depending of TRIGger:A:CAN:DLENgth (see commad below) setting the number of characters is fixed, all bytes need to be complete.

Parameters:

<Data>	01X-string with up to 64 character

Example: TRIG:A:CAN:DATA „10010110"

TRIGger:A:CAN:DLENgth <DataLength>
Specifies the data length for the CAN trigger.

Parameters:

<DataLength>	<numeric_value>	
	Min.:	0
	Max.:	8
	Inc.:	1
	Unit:	Byte
	*RST:	1

TRIGger:A:CAN:ACKerror <AcknowledgeError>
Specifies the trigger on acknowledge error when trigger type is set to error, using TRIGger:A:CAN:TYPE

Parameters:

<AcknowledgeError>	ON | OFF	
	*RST:	OFF

TRIGger:A:CAN:CRCerror <CRCerror>
Specifies the trigger on checksum error when trigger type is set to error, using TRIGger:A:CAN:TYPE .

Parameters:

<CRCerror>	ON | OFF	
	*RST:	OFF

TRIGger:A:CAN:FROMerror <FormError>
Specifies the trigger on form error when trigger type is set to error, using TRIGger:A:CAN:TYPE .

Parameters:

<FormError>	ON | OFF	
	*RST:	OFF

TRIGger:A:CAN:BITSterror <BitStuffingError>
Specifies the trigger on stuff bit error when trigger type is set to error, using TRIGger:A:CAN:TYPE .

Parameters:

<BitStuffingError>	ON | OFF	
	*RST:	ON

BUS<b>:CAN:FCOunt? <FrameCount>
Returns the number of frames.

Suffix:
<b> 1, 2
Select the bus.

Return values:
<FrameCount> Total number of decoded frames.

Usage: Query only

BUS<b>:CAN:FRAMe<n>:STATus? <FrameStatus>
Returns the status of frames.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<State> OK | BTST | CRC | FORM | NACK | INSufficient

OK	frame is valid
BTST	bit stuff error occured
CRC	cyclic redundancy check failed
FORM	wrong CRC or ACK delimiter or end of frame occured
NACK	acknowledge is missing
INSufficient	frame is incomplete

Usage: Query only

BUS<b>:CAN:FRAME<n>:STARt? <StartTime>
Returns the start time of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<StartTime>

Range:	depends on sample rate, record length, and time base
Increment:	depends on the time base
Default unit:	s

Usage: Query only

BUS<b>:CAN:FRAME<n>:STOP? <StopTime>
Returns the stop time of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<StopTime> Range: depends on sample rate, record length, and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<b>:CAN:FRAME<n>:TYPE? <FrameType>
Returns the frame type.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<FrameType> DATA | REMote | ERR | OVLD

DATA frame type is data
REMote frame type is remote
ERR frame type is error
OVLD frame type is overload

Usage: Query only

BUS<b>:CAN:FRAME<n>:DATA? <FrameData>
Returns a comma separated list of decimal values.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<FrameData> Comma-separated list of decimal values of data bytes of a frame

Usage: Query only

BUS<b>:CAN:FRAME<n>:ACKState? <AcknowledgeState>
Returns the status of the acknowledge field of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<AcknowledgeState> OK | UNDF

OK acknowledge state is o.k.
UNDF acknowledge state is undefined.

Usage: Query only

BUS<b>:CAN:FRAME<n>:CSState <ChecksumState>
Returns the status of the checksum field of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<ChecksumState> OK | UNDF

OK acknowledge state is o.k.
UNDF acknowledge state is undefined.

Usage: Query only

BUS<b>:CAN:FRAME<n>:DLCState? <DataLengthCodeState>
Returns the status of the data length code of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<DataLengthCodeState> OK | UNDF

OK data length code state is o.k.
UNDF data length code state is undefined.

Usage: Query only

BUS<b>:CAN:FRAME<n>:IDSTate? <IdentifierState>
Returns the status of the identifier field of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<IdentifierState> OK | UNDF

OK identifier state is o.k.
UNDF identifier state is undefined.

Usage: Query only

BUS<b>:CAN:FRAME<n>:ACKValue? <AcknowledgeValue>
Returns the value of the acknowledge bit of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<AcknowledgeValue> Decimal value

Usage: Query only

BUS<b>:CAN:FRAME<n>:CSValue? <ChecksumValue>
Returns the value of the checksum of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<ChecksumValue> Decimal value

Usage: Query only

BUS<**b**>**:CAN:FRAME**<**n**>**:DLCValue** <DataLengthCodeValue>
Returns the value of the data length code field of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<DataLengthCodeValue> Non-negative integer

Usage: Query only

BUS<**b**>**:CAN:FRAME**<**n**>**:IDTYpe?** <IdentifierType>
Returns the identifier type of the selected frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<IdentifierType> ANY | B11 | B29

ANY identifier type is any
B11 identifier type is 11 bit
B29 identifier type is 29 bit

Usage: Query only

BUS<**b**>**:CAN:FRAME**<**n**>**:IDValue?** <IdentifierValue>
Returns the value of the identifier of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<IdentifierValue> Decimal value

Usage: Query only

BUS<b>:CAN:FRAME<n>:BSEPosition? <BitStuffingErrorPosition>
Returns the time of the first bit stuffing error within the specified frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<BitStuffingErrorPosition> Unit: s

*RST: 0

Usage: Query only

BUS<b>:CAN:FRAME<n>:BCOunt? <ByteCount>
Returns the number of data bytes of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<ByteCount> Number of bytes

Usage: Query only

BUS<b>:CAN:FRAME<n>:BYTE<o>:STATe? <ByteStatus>
Returns the state of the a data byte of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

<o> *
Selects the byte number.

Return values:
<ByteStatus> OK | UNDF

OK data byte state is o.k.
UNDF data byte state is undefined.

Usage: Query only

BUS<**b**>**:CAN:FRAME**<**n**>**:BYTE**<**o**>**:VALue?** <ByteValue>
Returns the value of the data byte of a frame.

Suffix:

<b>	1, 2 Select the bus.
<n>	* Selects the frame.
<o>	* Selects the byte number.

Return values:

<ByteValue>	Decimal value

Usage: Query only

2.11.8 LIN

LIN - Configuration

Trigger

Decode

BUS<b>:LIN:DATA:SOURce <Source>
Sets the input channel to which the data line is connected.

Suffix:

<b> 1, 2
Select the bus.

Parameters:

<Source> CH1 | CH2 | CH3 | CH4 | D0 .... D15

*RST: CH1

BUS<b>:LIN:POLarity <Polarity>
Sets the polarity of the LIN.

Suffix:

<b> 1, 2
Select the bus.

Parameters:

<Polarity> IDLHigh | IDLLow

*RST: POS

BUS<b>:LIN:STANdard <Standard>
Sets the standard of LIN protocol.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<Standard> V1X | V2X | J2602 | AUTO

V1X	LIN Standard V1.x
V2X	LIN Standard V2.x
J2602	LIN Standard J206 (sets also BitRate to 10.417 KBit/s)
AUTO	any standard

*RST: V1X

BUS<b>:LIN:BITRate <BitRate>
Sets the bit rate for the LIN Decoder.

Suffix:
<b> 1, 2
Select the bus.

Parameters:
<BitRate> <numeric_value>

HMO35xx:	100Bit/s to 4.03MBit/s
HMO2524:	100Bit/s to 2.52MBit/s
HMO72x...202x:	100Bit/s to 2.016MBit/s
Default unit:	Bit/s

*RST: 9.6E03

TRIGger:A:LIN:TYPE <TriggerType>
Specifies the trigger type for LIN.

Parameters:
<TriggerType> SYNC | WKFRame | ID | IDDT | ERRCondition

SYNC	Sets the trigger to the synchronisation.
WKFRame	Sets the trigger to the wake up frame.
ID	Sets the trigger to the ID of a frame.
IDDT	Sets the trigger to the ID and data of a frame.
ERRCondition	Sets the trigger to the error condition of a frame.

*RST: SYNC

TRIGger:A:LIN:ICONdition <IdentifierCondition>
Specifies the condition for the identifier.

Parameters:
<IdentifierCondition> EQUual | NEQual | GTHan | LTHan

EQUual	Sets the condition for identifier to equal.
NEQual	Sets the condition for identifier to not equal.
GTHan	Sets the condition for identifier to greater then.
LTHan	Sets the condition for identifier to less then.

*RST: EQU

TRIGger:A:LIN:IDENtifier <Identifier>
Specifies the identifier, only 6 character are allowed.

Parameters:
<Identifier> 01X-string with up to 6 character

Example: `TRIG:A:LIN:IDEN "100001"`

TRIGger:A:LIN:DCONdition <DataCondition>
Specifies the condition for the data.

Parameters:
<DataCondition> EQUual | NEQual | GTHan | LTHan

EQUual	Sets the condition for identifier to equal.
NEQual	Sets the condition for identifier to not equal.
GTHan	Sets the condition for identifier to greater then.
LTHan	Sets the condition for identifier to less then.

*RST: EQU

TRIGger:A:LIN:DATA <Data>
Specifies the data for LIN trigger. Depending of `TRIGger:A:LIN:DLENgth` (see command below) setting the number of characters is fixed, all bytes need to be complete.

Parameters:
<Data> 01X-string with up to 64 character

Example: `TRIG:A:LIN:DATA „10100101"`

TRIGger:A:LIN:DLENgth <DataLength>
Specifies the data length for the LIN trigger.

Parameters:
<DataLength>

Range:	1 to 8
Increment:	1
Default unit:	Byte

*RST: 1

TRIGger:A:LIN:CHKSerror <ChecksumError>
Specifies the trigger on checksum error when trigger type is set to error condition, using `TRIGger:A:LIN:TYPE` <TriggerType>.

Parameters:
<ChecksumError> ON | OFF

*RST: ON

TRIGger:A:LIN:IPERror <IdentifierParityError>
Specifies the trigger on parity error on identifier when trigger type is set to error condition, using `TRIGger:A:LIN:TYPE` .

Parameters:
<IdentifierParityError> ON | OFF

*RST: OFF

TRIGger:A:LIN:SYERror <SynchronisationError>
Specifies the trigger on synchronisation error when trigger type is set to error, using `TRIGger:A:LIN:TYPE` .

Parameters:

<SynchronisationError>	ON | OFF	
	*RST:	OFF

BUS<**b**>**:LIN:FCOunt?** <FrameCount>
Returns the number of received frames.

Suffix:

<b>	1, 2 Select the bus.

Return values:

<FrameCount>	Total number of decoded frames.

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:STATus?** <FrameStatus>
Returns the status of frames.

Suffix:

<b>	1, 2 Select the bus.
<n>	* Selects the frame.

Return values:

<State>	OK | UART | CHCK | PRER | WAK | SYER | INS | ERR | LEN	
	OK	frame is valid
	UART	
	CHCK	checksum error occured
	PRERror	parity error
	WAKeup	wake up occured
	SYER	synchronisation error
	INSufficient	identifier is insufficient
	ERRor	error
	LENer	length error

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:STARt?** <StartTime>
Returns the start time of a frame.

Suffix:

<b>	1, 2 Select the bus.
<n>	* Selects the frame.

Return values:

<StartTime>	Range:	depends on sample rate, record length, and time base
	Increment:	depends on the time base
	Default unit:	s

Usage: Query only

BUS<b>:LIN:FRAME<n>:STOP? <StopTime>
Returns the stop time of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<StopTime> Range: depends on sample rate, record length, and time base
Increment: depends on the time base
Default unit: s

Usage: Query only

BUS<b>:LIN:FRAME<n>:VERSion? <FrameVersion>
Returns the version of LIN.

Suffix:
<b> 1, 2
Select the bus.
<n> *
Selects the frame.

Return values:
<FrameVersion> V1X | V2X | UNK

V1X LIN version is 1.x
V2X LIN version is 2.x
UNK LIN version is unknown

Usage: Query only

BUS<b>:LIN:FRAME<n>:DATA? <FrameData>
Returns a comma separated list of decimal values of the data bytes.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<FrameData> Comma-separated list of decimal values of data bytes of a frame

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:IDSTate?** <IdentifierState>
Returns the status of the identifier field of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<IdentifierState> OK | PRERror | UVAL | INSufficient

OK	identifier state is o.k.
PRERror	parity error at the identifier
UVAL	address is unknown
INSufficient	identifier is insufficient

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:IDValue?** <IdentifierValue>
Returns the value of the identifier of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<IdentifierValue> Decimal value

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:IDPValue?** <IdentifierParityValue>
Returns the value of the parity of the identifier of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<IdentifierParityValue> Decimal value

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:SYSTate?** <SyncFieldState>
Returns the status of synchronization field of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<SyncFieldState> OK | ERR | UNDF

OK	sync field state is o.k.
ERR	sync error.
UNDF	sync field state is undefined.

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:CSSTate?** <ChecksumState>
Returns the state of the checksum of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<ChecksumState> OK | ERR | UNDF

OK	checksum field state is o.k.
ERR	checksum error.
UNDF	checksum state is undefined.

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:CSValue?** <ChecksumValue>
Returns the value of the checksum of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<ChecksumValue> Decimal value

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:BCOunt?** <ByteCount>
Returns the number of bytes of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

Return values:
<ByteCount> Number of words (bytes)

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:BYTE**<**o**>**:STATe?** <ByteStatus>
Returns the status of a data byte of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

<o> *
Selects the byte number.

Return values:
<ByteStatus> OK | INS | UART

OK byte status is o.k.
INS byte status is insufficient.
UART error within the byte.

Usage: Query only

BUS<**b**>**:LIN:FRAME**<**n**>**:BYTE**<**o**>**:VALue** <ByteValue>
Returns the value of the byte of a frame.

Suffix:
<b> 1, 2
Select the bus.

<n> *
Selects the frame.

<o> *
Selects the byte number.

Return values:
<ByteValue> Decimal value

Usage: Query only

2.12 Data and File Management

This chapter describes commands that store or restore data and measurement results.

2.12.1 Output Control

HCOPy:DESTination <Medium>
Defines whether the screenshot is saved or printed.

Parameters:

<Medium> „MMEM" | „SYST:COMM:PRIN"

„MMEM"
Saves the screenshot to a file. Specify the file name and location with `MMEMory:NAME`.

„SYST:COMM:PRIN"
Prints on the printer specified with `SYSTem:COMMunicate:PRINter:SELect`.
The printer must be specified before the `HCOPy:DESTination` is sent.

*RST: „MMEM"

MMEMory:NAME <FileName>
Defines the file name to store an image of the display with `HCOPy[:IMMediate]`.

Parameters:

<FileName> String parameter

HCOPy[:IMMediate]
Prints an image of the display to the printer or saves an image to a file, depending on the `HCOPy:DESTination` setting.

The printer is defined by `SYSTem:COMMunicate:PRINter:SELect`.

The file name for storage is defined by `MMEMory:NAME`.

Usage: Event

HCOPy:LANGuage <Format>
Defines the format of the printed or saved screenshot.

Parameters:

<Format> GDI | BMP | PNG | GIF

GDI For output on printer
BMP | PNG | GIF File formats for saved screenshots

*RST: PNG

HCOPy:PAGE:SIZE <Size>
Defines the page size to be used.

Parameters:
<Size> A4 | A5 | B5 | B6 | EXECutive

HCOPy:PAGE:ORIentation <Orientation>
Defines the page orientation.

Parameters:
<Orientation> LANDscape | PORTrait

HCOPy:COLOR:SCHeme <ColorScheme>
Defines the color mode for saved and printed screenshots.

Parameters:
<ColorScheme> COLor | GRAYscale | INVerted
INVerted inverts the colors of the output, i.e. a dark waveform is printed on a white background.

*RST: COLor

SYSTem:COMMunicate:PRINter:SELect <PrinterName>
Selects a configured printer.

Parameters:
<PrinterName> String parameter
Enter the string as it is returned with
`SYSTem:COMMunicate:PRINter:ENUMerate:FIRSt?` or
`SYSTem:COMMunicate:PRINter:ENUMerate[:NEXT]?.`

SYSTem:COMMunicate:PRINter:ENUMerate:FIRSt?
Queries the name of the first printer in the list of printers. The names of other installed printers can be queried with the `SYSTem:COMMunicate:PRINter:ENUMerate[:NEXT]?` command.

Return values:
<PrinterName> String parameter
If no printer is configured an empty string is returned.

Usage: Query only

SYSTem:COMMunicate:PRINter:ENUMerate[:NEXT]?
Queries the name of the next printer installed. The `SYSTem:COMMunicate:PRINter:ENUMerate:FIRSt?` command should be sent previously to return to the beginning of the printer list and query the name of the first printer.

Return values:
<PrinterName> String parameter
After all available printer names have been returned, an empty string enclosed by quotation marks („) is returned for the next query. Further queries are answered by a query error.

Usage: Query only

2.12.2 MMEMory Commands

The Mass MEMomory subsystem provides commands to access the storage media and to save and reload instrument settings.

The oscilloscope has three storage devices indicated as drives:
- /INT:internal storage with default directories for each data type
- /USB_FRONT:USB connector on the front panel
- /USB_REAR:USB connector on the rear panel

Common computer and network drives like C:, D:, \\server\share are not available.

Name conventions

The names of files and directories have to meet the following rules:
- Only the 8.3 format with ASCII characters is supported.
- No special characters are allowed.
- Use / (slash) instead of \ (backslash).

MMEMory:DRIVes?

Returns the storage devices available on the oscilloscope.

Return values:

<Drive> List of strings, for example, "/INT", "/USB_FRONT", "/USB_REAR"

/INT:	internal storage
/USB_FRONT:	USB connector on the front panel
/USB_REAR:	USB connector on the rear panel

Usage: Query only

MMEMory:MSIS [<MassStorageIS>]
Changes the storage device (drive).

Parameters:

<MassStorageIS> One of the available drives
/INT, /USB_FRONT, or /USB_REAR

Example:

```
MMEM:MSIS "/USB_FRONT"
```

Sets USB stick connected to the front panel as storage device to be used.

MMEMory:DCATalog? <PathName>
Returns the subdirectories of the specified directory. The result corresponds to the number of strings returned by the `MMEMory:DCATalog:LENgth?` command.

Query parameters:

<PathName> — String parameter
Specifies the directory.

Return values:

<FileEntry> — String parameter
List of subdirectory strings separated by commas. If the specified directory does not have any subdirectory, the current and the parent directories are returned ("..,,0", "...,0")

Example: Query for directories with absolute path:
```
MMEM:DCAT? "/USB_FRONT/*"
received ".,,0",".,,0","DATA,,0","DATA_NEW,,0","SCREENSHOTS,,0"
MMEM:DCAT:LENG? „/USB_FRONT/*"
received 5
```

Example: Query for directories in the current directory:
```
MMEM:CDIR "/USB_FRONT/DATA/"
MMEM:DCAT? "*"
received ".,,0",".,,0","JANUARY,,0","FEBRUARY,,0"
MMEM:DCAT:LENG? "*"
received 4
```

Example: Query with filter:
```
MMEM:DCAT? „/USB_FRONT/DA*"
received "DATA,,0","DATA_NEW,,0"
MMEM:DCAT:LENG? "/USB_FRONT/DA*"
received 2
```

Usage: Query only

MMEMory:DCATalog:LENGth? <PathName>
Returns the number of directories in specified directory. The result corresponds to the number of strings returned by the `MMEMory:DCATalog?` command above.

Return values:

<FileEntryCount> — Number of directories.

Query Parameters:

<PathName> — String parameter
Specifies the directory.

Example:
```
MMEM:DCAT:LENG? "/USB_FRONT"
--> 2
```

Usage: Query only

MMEMory:MDIRectory <DirectoryName>
Creates a new directory with the specified name.

Setting parameters:

<DirectoryName> String parameter
Absolute path including the storage device, or relative to the current directory.

Example: Create directory DATA on the front USB flash device, with absolute path:
`MMEM:MDIR "/USB_FRONT/DATA"`

Example: Create directory JANUARY in the DATA directory, with relative path:
`MMEM:CDIR "/USB_FRONT/DATA/"`
`MMEM:MDIR "JANUARY"`

Usage: **Setting only**

MMEMory:CDIRectory [<DirectoryName>]
Changes the default directory for file access.

Setting Parameters:

<DirectoryName> String parameter to specify the directory.
If the string also contains the storage device, the command `MMEM:MSIS` is executed implicitely.

Example: `MMEM:CDIR "/USB_FRONT/DATA"`

MMEMory:RDIRectory <DirectoryName>
Deletes the specified directory.
Note: All subdirectories and all files in the specified directory and in the subdirectories will be deleted!

You cannot delete the current directory or a superior directory. In this case, the instrument returns an execution error.

Setting parameters:

<DirectoryName> String parameter, absolute path or relative to the current directory

Example: `MMEM:RDIR "/INT/TEST"`
Deletes the directory TEST in the internal storage device, and all files and subdirectories in the directory.

Usage: Setting only

MMEMory:CATalog? <PathName>[,<Format>]
Returns the a list of files contained in the specified directory. The result corresponds to the number of files returned by the `MMEMory:CATalog:LENgth?` command.

Query parameters:

<PathName> String parameter
Specifies the directory. A filter can be used to list, for example, only files of a given file type.

<Format> ALL | WTIMe

ALL: Extended result including file, date, time and attributes
WTIMe: Result including file, date, time

Return values:

<UsedMemory> Total amount of storage currently used in the directory, in bytes.

<FreeMemory> Total amount of storage available in the directory, in bytes.

<FileEntry> String parameter
All files of the directory are listed with their file name, format and size in bytes.

Example: Query for files in the DATA directory, with absolute path:

```
MMEM:CAT? "/USB_FRONT/DATA/*.*"
received: 511104,8633856,"MONDAY.TXT,,8","TUESDAY.CSV,,8"
```

Example: Query for TXT files in the DATA directory, with relative path:

```
MMEM:CDIR "/USB_FRONT/DATA"'
MMEM:CAT? "*.TXT"
received: 511104,8633856,"MONDAY.TXT,,8"
MMEM:CAT:LENGTH? "*.TXT"
received 1
```

Usage: Query only

MMEMory:CATalog:LENGth? <PathName>
Returns the number of files in the specified directory. The result corresponds to the number of files returned by the `MMEMory:CATalog?` command above.

Return values:
<Count> Number of files

Query Parameters:
<PathName> String parameter
Directory to be queried

Usage: Query only

MMEMory:COPY <FileSource>,<FileDestination>
Copies data to another directory on the same or different storage device. The file name can be changed, too.

Setting Parameters:
<FileSource> String parameter
Name and path of the file to be copied

<FileDestination> String parameter
Name and path of the new file. If the file already exists, it is overwritten without notice.

Example:

```
MMEM:COPY "/INT/SETTINGS/SET001.SET",
"/USB_FRONT/SETTINGS/TESTSET1.SET"
```

Usage: Setting only

MMEMory:MOVE <**FileSource**>,<**FileDestination**>
Moves an existing file to a new location.

Setting Parameters:
<FileSource> String parameter
Path and name of the file to be moved

<FileDestination> String parameter
Path and name of the new file

Example:

```
MMEM:MOVE "/INT/SETTINGS/SET001.SET",
"/USB_FRONT/SETTINGS"
```

Usage: Setting only

MMEMory:DELete <FileSource>
Removes a file from the specified directory.

Setting Parameters:
<FileSource> String parameter
File name and path of the file to be removed. If the path is omitted, the specified file will be deleted in the current directory. Filters are not allowed.

Usage: Setting only

MMEMory:DATA <FileName>,<Data>
Writes data to the specified file in the current directory `MMEMory: CDIRectory` , or reads the data.

Parameters:
<Data> 488.2 block data
The block begins with character ‚#'. The next digit is the length of the length information, followed by this given number of digits providing the number of bytes in the binary data attached.

Parameters for setting and query:
<FileName> String parameter containing the file name

Example:
```
MMEM:DATA "abc.txt", #216This is the file
```
#2: the length infomation has two digits
16: the binary data has 16 bytes

```
MMEM:DATA? "abc.txt"
```
received: `This is the file`

MMEMory:STORe:STATe <StateNumber>,<FileName>
Saves the current device settings to the specified file in the current directory.

Setting parameters:
<StateNumber> Range: 1 to 1
Increment: 0
*RST: 1

<FileName> String parameter
File name, with or without file extension

Example:
```
MMEM:CDIR "/USB_FRONT/DATA"'
MMEM:STOR:STAT 1,"MORNING.SET"
```

Usage: Setting only

MMEMory:LOAD:STATe <StateNumber>,<FileName>
Loads the device settings from the specified file in the current directory.

Setting parameters:
<StateNumber> Range: 1 to 1
Increment: 0
*RST: 1

<FileName> String parameter
File name, with or without file extension

Example:
```
MMEM:CDIR "/USB_FRONT/DATA"'
MMEM:LOAD:STAT 1,"MORNING"
```

Usage: Setting only

2.13 General Instrument Setup

DISPlay:LANGuage <Language>
Sets the language in which the softkey labels, help and other screen information can be displayed. Supported languages are listed in the „Specifications" data sheet.

Parameters:

<Language>	ENGLish | GERMan | FRENch | SPANish | SCHinese | TCHinese
	SCHinese: Simplified Chinese
	TCHinese: Traditional Chinese

DISPlay:LANGuage:ADD <Language>,<FilePath>
Adds the selected language to the device including softkey labels and help. To change the displayed language, use `DISPlay: LANGuage` .

Parameters:

<Language>	ENGLish | GERMan | FRENch | SPANish | SCHinese | TCHinese
<FilePath>	String parameter contains the source path and the filename of the language installation file.
Example:	`DISPlay:LANGuage:ADD FRENch,"/USB_FRONT/HMO_LN00.HLU"`
	Loads the language installation file HMO_LN00.HLU from the USB device on the front panel and adds the french language.
Usage:	Setting only

DISPlay:LANGuage:REMove <Language>
Removes the selected language from the device including softkey labels and help.

Parameters:

<Language>	ENGLish | GERMan | FRENch | SPANish | SCHinese | TCHinese
Example:	`DISPlay:LANGuage:REMove FRENch`
	Can be used to remove french language from the device.
Usage:	Setting only

DISPlay:LANGuage:CATalog? <Languages>
Returns a list of available languages of the device.

Return values:
<Languages> List of available languages.

Example: `DISPlay:LANGuage:CAT?`
Returns ENGL,GERM,FREN, if english, german and french languages are installed.

Usage: Query only.

SYSTem:NAME <Name>
Defines an instrument name.

Parameters:
<Name> String with max. 20 characters

Example: `SYST:NAME "MyScope"`

SYSTem:DATE <Year>,<Month>,<Day>
Specifies the internal date for the instrument.

Parameters:
<Year> Increment: 1

<Month> Range: 1 to 12
Increment: 1

<Day> Range: 1 to 31
Increment: 11

Example: `SYSTem:DATE 2012,10,1` (Set the device date to october 1st in the year 2012)

`SYSTem:DATE?` (returns `2012,10,1`)

Usage: SCPI confirmed

SYSTem:TIME <Hour>,<Minute>,<Second>
Specifies the internal time for the instrument.

Parameters:
<Hour> Range: 0 to 23
Increment: 1
Default unit: h

<Minute> Range: 0 to 59
Increment: 1
Default unit: min

<Second> Range: 0 to 59
Increment: 1
Default unit: s

Example: `SYSTem:TIME 12,15,0`
Set the time to quarter past twelve.

`SYSTem:TIME?`
returns `12,15,0`

Usage: SCPI confirmed

SYSTem:TREE?
Returns a list of implemented remote commands.

SYSTem:SET <Setup>
Defines or queries the device settings that can be saved and load manually with SAVE/RECALL > „Device Settings".

Parameters:

<Setup>	488.2 block data

Usage: SCPI confirmed

SYSTem:ERRor:[NEXT]? <Error>
Queries the error/event queue for the oldest item and removes it from the queue. The response consists of an error number and a short description of the error.

Positive error numbers are instrument-dependent. Negative error numbers are reserved by the SCPI standard.

Return values:

<Error>	Error/event_number,"Error/event_description>[;Devicedependent info]" If the queue is empty, the response is 0,"No error"
Usage:	Query only SCPI confirmed

SYSTem:ERRor:ALL? <Error>
Queries the error/event queue for all unread items and removes them from the queue.
The response is a comma separated list of error number and a short description of the error in FIFO order.

Positive error numbers are instrument-dependent. Negative error numbers are reserved by the SCPI standard.

Return values:

<Error>	List of:Error/event_number,"Error/event_description>[;Devicedependent info]" If the queue is empty, the response is 0,"No error"
Usage:	Query only SCPI confirmed

SYST:PRESet
Resets the instrument to the default state, has the same effect as *RST.

Usage: Event

2.14 Status Reporting

2.14.1 STATus:OPERation Register

The commands of the `STATus:OPERation` subsystem control the status reporting structures of the STATus:OPERation register:

See also:
– chapter 1.6.1, „Structure of a SCPI Status Register“, on page 18
– „STATus:OPERation Register“, on page 22

The following commands are available:

STATus:OPERation:CONDition? <Condition>
Returns the of the CONDition part of the operational status register.

Return values:

<Condition> Condition bits in decimal representation. ALIGnment (bit 0) , SELFtest (bit 1) , AUToset (bit 2), WTRigger (bit 3).
Range: 1 to 65535
Increment: 1

Usage: Query only

STATus:OPERation:ENABle <Enable>

Parameters:

<Enable>
Range: 1 to 65535
Increment: 1

STATus:OPERation:NTRansition <NegativeTransition>

Parameters:

<NegativeTransition>
Range: 1 to 65535
Increment: 1

STATus:OPERation:PTRansition <PositiveTransition>

Parameters:

<PositiveTransition>
Range: 1 to 65535
Increment: 1

STATus:OPERation[:EVENt]? <Event>

Return values:

<Event>
Range: 1 to 65535
Increment: 1

Usage: Query only

2.14.2 STATus:QUEStionable Registers

The commands of the `STATus:QUEStionable` subsystem control the status reporting structures of the STATus:QUEStionable registers:

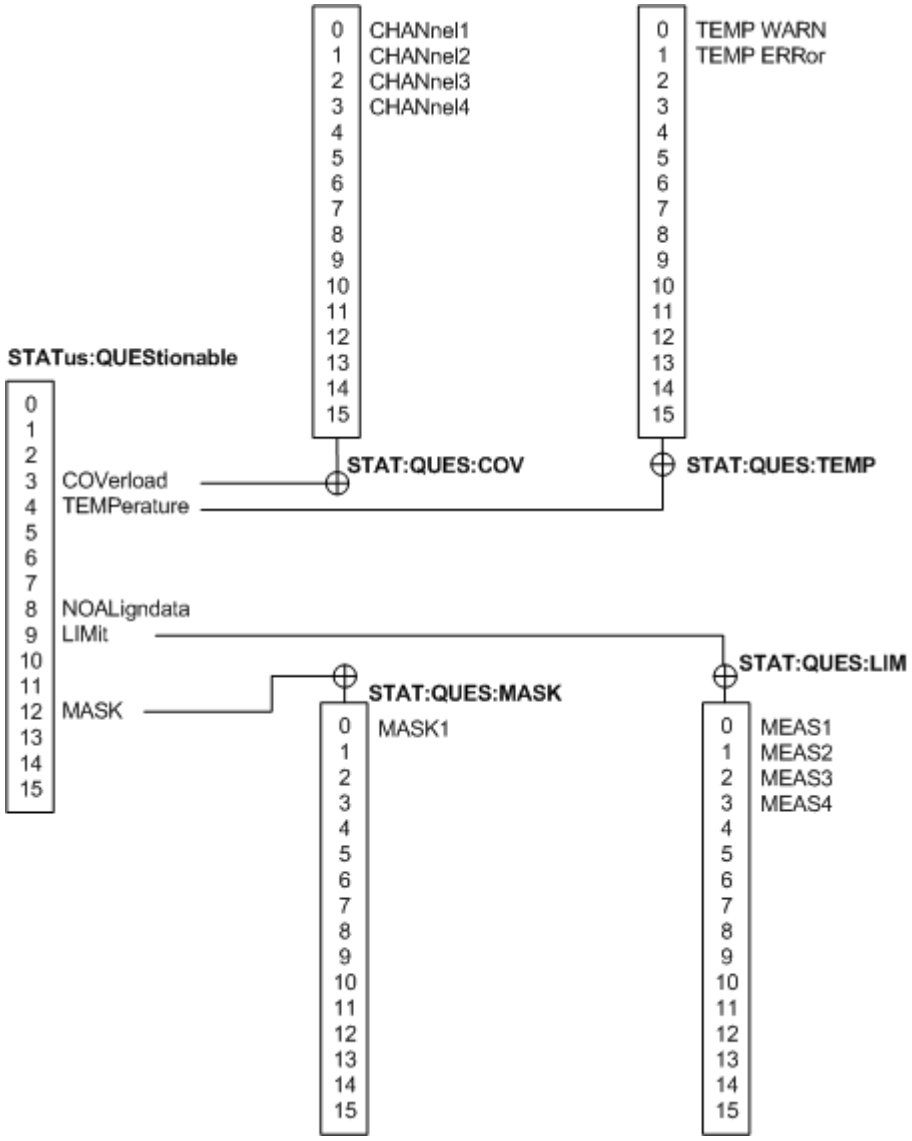


Fig. 2.1:Structure of the STATus:QUEStionable register

See also:

- chapter 1.6.1, „Structure of a SCPI Status Register", on page 18
- „STATus:QUEStionable Register", on page 22

The following commands are available:

STATus:PRESet
Resets all STATUS:QUESTIONALBLE registers.

Usage: Event

STATus:QUEStionable:CONDition?
STATus:QUEStionable:COVerload:CONDition?
STATus:QUEStionable:LIMit:CONDition?
STATus:QUEStionable:MASK:CONDition?

Returns the contents of the CONDition part of the status register to check for questionable instrument or measurement states. Reading the CONDition registers does not delete the contents.

Return values:
<Condition> Condition bits in decimal representation
Range: 1 to 65535
Increment: 1

Usage: Query only

STATus:QUEStionable:ENABle <Enable>
STATus:QUEStionable:COVerload:ENABle <Enable>
STATus:QUEStionable:LIMit:ENABle <Enable>
STATus:QUEStionable:MASK:ENABle <Enable>

Sets the enable mask that allows true conditions in the EVENt part to be reported in the summary bit. If a bit is set to 1 in the enable part and its associated event bit transitions to true, a positive transition occurs in the summary bit and is reported to the next higher level.

Parameters:
<Enable> Bit mask in decimal representation
Range: 1 to 65535
Increment: 1

Example:
```
STATus:QUEStionable:MASK:ENABle 24
```
Set bits no. 3 and 4 of the STATus:QUEStionable:MASK:ENABle register part:24 = 8 + 16 = 23 + 24

STATus:QUEStionable[:EVENt]?
STATus:QUEStionable:COVerload[:EVENt]?
STATus:QUEStionable:LIMit[:EVENt]?
STATus:QUEStionable:MASK[:EVENt]?

Returns the contents of the EVENt part of the status register to check whether an event has occurred since the last reading. Reading an EVENt register deletes its contents.

Return values:

<Event> Event bits in decimal representation
Range: 1 to 65535
Increment: 1

Usage: Query only

STATus:QUEStionable:NTRansition <NegativeTransition>
STATus:QUEStionable:COVerload:NTRansition <NegativeTransition>
STATus:QUEStionable:LIMit:NTRansition <NegativeTransition>
STATus:QUEStionable:MASK:NTRansition <NegativeTransition>

Sets the negative transition filter. If a bit is set, a 1 to 0 transition in the corresponding bit of the condition register causes a 1 to be written in the corresponding bit of the event register.

Parameters:

<NegativeTransition> Bit mask in decimal representation
Range: 1 to 65535
Increment: 1

Example:

```
STATus:QUEStionable:MASK:NTRansition 24
```

Set bits no. 3 and 4 of the STATus:QUEStionable:MASK:NTRansition register part:24 = 8 + 16 = 23 + 24

STATus:QUEStionable:PTRansition <PositiveTransition>
STATus:QUEStionable:COVerload:PTRansition <PositiveTransition>
STATus:QUEStionable:LIMit:PTRansition <PositiveTransition>
STATus:QUEStionable:MASK:PTRansition <PositiveTransition>

Sets the positive transition filter. If a bit is set, a 0 to 1 transition in the corresponding bit of the condition register causes a 1 to be written in the corresponding bit of the event register.

Parameters:

<PositiveTransition> Bit mask in decimal representation
Range: 1 to 65535
Increment: 1

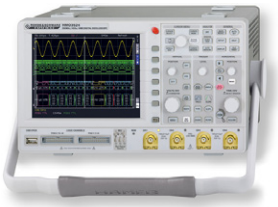
Example:

```
STATus:QUEStionable:MASK:PTRansition 24
```

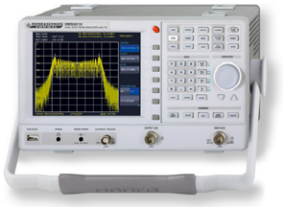
Set bits no. 3 and 4 of the STATus:QUEStionable:MASK:PTRansition register part:24 = 8 + 16 = 23 + 24

HAMEG®
Instruments
A Rohde&Schwarz Company

Oscilloscopes



Spectrum Analyzer



Power Supplies

Modular System
Series 8000

Programmable Instruments
Series 8100

authorized dealer

www.hameg.com

Release: Oktober 2012
Subject to change without notice

A Rohde & Schwarz Company
DQS-Certification:DIN EN ISO 9001
Reg.-Nr.:071040 QM

HAMEG Instruments GmbH
Industriestraße 6
D-63533 Mainhausen
Tel +49 (0) 61 82 800-0
Fax +49 (0) 61 82 800-100
sales@hameg.com