Programmer's Manual v2.2 (August 2017)

Phase Noise Measurement Systems

Berkeley Nucleonics Corporation - 2955 Kerner Blvd. - San Rafael, CA 94901

800-235-7858 or LiveChat @ www.berkeleynucleonics.com

WARRANTY

All BNC instruments are warranted against defects in material and workmanship for a period of two years from the date of shipment. BNC will, at its option, repair or replace products that prove to be defective during the warranty period, provided they are returned to BNC and provided the preventative maintenance procedures are followed. Repairs necessitated by misuse of the product are not covered by this warranty. No other warranties are expressed or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose. BNC is not liable for consequential damages.

IMPORTANT! PLEASE READ CAREFULLY

Copyright

Table of Contents

1 Introduction

This manual provides information for remote operation of the PNT Signal Source Analyzers using commands sent from an external controller. It includes the following:

- A general description of the LAN and the bus data transfer and control functions
- A general description of how to establish connection to the PNT via LAN or USB
- A listing of the IEEE-488 Interface Function Messages recognized by the instrument with a description of its response
- A complete listing and description of all the Standard Commands for Programmable Instruments (SCPI) commands that can be used to control instrument operation with examples of command usage

2 Programming the PNT

The PNT can be accessed through LAN or USB interface. All interfaces use standard SCPI command set to pass commands to the device. GPIB interface is also available optionally.

2.1 LAN

The PNT can be remotely programmed via a 10/100/1000Base-T LAN interface and LAN-connected computer using one of several LAN interface protocols. The LAN allows instruments to be connected together and controlled by a LAN-based computer. LAN and its associated interface operations are defined in the IEEE 802.2 standard.

The PNT support the following LAN interface protocols:

1) **Socket based LAN:** the application programming interface (API) provided with the instrument supports general programming using the LAN interface under Windows operating system.
2) **VXI-11**
3) **Telephone Network** (TELNET): TELNET is used for interactive, one command at a time instrument control.
4) **Internet protocol** optionally supported

For LAN operation, the instrument must be connected to the LAN, and an IP address must be assigned to the instrument either manually or by using DHCP client service. Your system administrator can tell you which method to use. (Most current LAN networks use DHCP.)

DHCP Configuration

If the DHCP server uses dynamic DNS to link the hostname with the assigned IP address, the hostname may be used in place of the IP address. Otherwise, the hostname is not usable.

2.2 Ethernet Interface Connection and Setup

The PNT fully supports the IEEE-802.3 standard. Most front panel functions (except power on/off) can be remotely controlled via a network server and an Ethernet connection. The PNT software supports the TCP/IP network protocol.

Ethernet uses a bus or star topologies where all of the interfacing devices are connected to a central cable called the bus, or are connected to a hub. Ethernet uses the CSMA/CD access method to handle simultaneous transmissions over the bus. CSMA/CD stands for *Carrier Sense Multiple Access/Collision Detection*. This standard enables network devices to detect simultaneous data

channel usage, called a *collision*, and provides for a contention protocol. When a network device detects a collision, the CSMA/CD standard dictates that the data will be retransmitted after waiting a random amount of time. If a second collision is detected, the data is again retransmitted after waiting twice as long. This is known as exponential back off.

The TCP/IP setup requires the following:

- IP Address: Every computer/electronic device in a TCP/IP network requires an IP address. An IP address has four numbers (each between 0 and 255) separated by periods.

  For example: 192.168.1.50 is a valid IP address.

- Subnet Mask: The subnet mask distinguishes the portion of the IP address that is the network ID from the portion that is the station ID. The subnet mask 255.255.0.0, when applied to the IP address given above, would identify the network ID as 192.168 and the station ID as 1.50. All stations in the same local area network should have the same network ID, but different station IDs.

- Default Gateway: A TCP/IP network can have a gateway to communicate beyond the LAN identified by the network ID. A gateway is a computer or electronic device that is connected to two different networks and can move TCP/IP data from one network to the other. A single LAN that is not connected to other LANs requires a default gateway setting of 0.0.0.0. If you have a gateway, then the default gateway would be set to the appropriate value of your gateway.

- MAC Address: A MAC address is a unique 48-bit value that identifies a network interface card to the rest of the network. Every network card has a unique MAC address permanently stored into its memory.

Interface between the instrument and other devices on the network is via a category five (CAT-5) interface cable connected to a network. This cable uses four twisted pairs of copper insulators terminated into an RJ45 connector. CAT-5 cabling is capable of supporting frequencies up to 100 MHz and data transfer speeds up to 1 Gbps, which accommodates 1000Base-T, 100Base-T, and 10Base-T networks.

The instrument can be remotely programmed using the VXI-11 protocol. A VISA I/O library (like NI-VISA™) is used on the server side to facilitate the communications. A VISA installation on the controller is a prerequisite for remote control over VXI-11 interface. VISA is a standardized software interface library providing input and output functions to communicate with instruments. For more information about VISA refer to the VISA library supplier’s documentation.

The SCPI command set listed in the PNT programmer’s manual applies to LAN programming as well.

Only the IP address or the device name is required for link setup. The IP address/device name is part of the "visa resource string" used by the programs for identification and control of the instrument. The visa resource string has the form:

TCPIP::ipaddr::inst0::INSTR

ipaddr has to be replaced by the IP address or the computer name of the instrument.

For instance, if the instrument has the IP address 192.168.1.50, *TCPIP::192.168.1.50::inst0::INSTR* is the valid resource name. Specification of **inst0** in the resource name is optional. In this example, also

TCPIP::192.168.1.50::INSTR is therefore a valid resource name.

TCPIP designates the network protocol used and **INSTR** indicates that the VXI-11 protocol is used. If several instruments are connected to the network, each instrument has its own IP address and associated resource name. The controller identifies these instruments by means of the resource name.

2.3 Using Sockets LAN

Sockets LAN is a method used to communicate with the instrument over the LAN interface using the Transmission Control Protocol/Internet Protocol (TCP/IP). A socket is a fundamental technology used for computer networking and allows applications to communicate using standard mechanisms built into network hardware and operating systems. The method accesses a port on the instrument from which bidirectional communication with a network computer can be established.

Sockets LAN can be described as an internet address that combines Internet Protocol (IP) with a device port number and represents a single connection between two pieces of software. The socket can be accessed using code libraries packaged with the computer operating system. Two common versions of socket libraries are the Berkeley Sockets Library for UNIX systems and Winsock for Microsoft operating systems.

Your instrument implements a socket Applications Programming Interface (API) that is compatible with Berkeley socket for UNIX systems, and Winsock for Microsoft systems. The instrument is also compatible with other standard sockets APIs. The instrument can be controlled using predefined SCPI functions once the socket connection is established in your program. Socket connection is available on **port 18**.

2.4 Using and Configuring VXI-11 (VISA)

The instrument supports the LAN interface protocol described in the VXI- 11 standard. VXI- 11 is an instrument control protocol based on Open Network Computing/Remote Procedure Call (ONC/RPC) interfaces running over TCP/IP.

A range of standard software such as NI-VISA or Agilent IO Config is available to setup the computer-instrument interface for the VXI- 11 protocol. Please refer to the applicable software user manual and documentation for information on running the program and configuring the VXI-11 interface. The program is used to configure the LAN client. Once the computer is configured for a LAN client, you can use the VXI- 11 protocol and the VISA library to send SCPI commands to the instrument over the LAN interface. Example programs are available on request under Info@berkeleynucleonics.com

VISA is an IO library used to develop IO applications and instrument drivers that comply with industry standards. It is recommended that the VISA library be used for programming the instrument. The NI-VISA and Agilent VISA libraries are similar implementations of VISA and have the same commands, syntax, and functions.

2.5 Using Telnet LAN (Port 18)

Telnet provides a means of communicating with the instrument over the LAN. The Telnet client, run on a LAN connected computer, will create a login session on the instrument. A connection, established between computer and instrument, generates a user interface display screen with “>” prompts on the command line.

Using the Telnet protocol to send commands to the instrument is similar to communicating with the instrument over LAN. You establish a connection with the instrument and then send or receive information using predefined commands. Communication is interactive: one command at a time. The telnet service is available on **port 18**.

Once a telnet session to the device is established, the echo can be enabled by typing

SYST:COMM:SOCK:ECHO ON

Following this command a prompt “>>” should become visible.

2.6 USB

There are two standard types of USB ports. The external controller (PC) must be connected via the USB host port (type A), while the PNT and other USB compatible devices must be connected via the USB interface port (type B)

The PNT support the following USB interface protocols:

1) **USBTMC class device via VISA**: **USBTMC** stands for **USB Test & Measurement Class**. USBTMC is a protocol built on top of USB that allows GPIB-like communication with USB devices. From the user's point of view, the USB device behaves just like a GPIB device. USBTMC allows instrument manufacturers to upgrade the physical layer from GPIB to USB while maintaining software compatibility with existing software, such as instrument drivers and any application that uses VISA. This is also what the VXI-11 protocol provides for TCP/IP.
2) **USBTMC class device with IVI host drivers** the application programming interface (API) provided with the instrument supports general programming using the USB interface under Windows operating system.

2.7 USB-TMC Interface Connection and Setup using VISA

The USB (Universal Serial Bus) remote control system provides device control via USB, which is equivalent to control via *GPIB*. Connection is made through an interface in compliance with USBTMC-USB488 and USB 2.0.

USBTMC stands for USB Test & Measurement Class. USBTMC is a protocol built on top of USB that allows GPIB-like communication with USB devices. From the user's point of view, the USB device behaves just like a GPIB device. For example, you can use VISA Write to send the *IDN? query and use VISA Read to get the response. The USBTMC protocol supports service request, triggers and other GPIB specific operations.

USBTMC upgrades the physical layer from GPIB to USB while maintaining software compatibility with existing software, such as instrument drivers and any application that uses VISA. This is also what the VXI-11 protocol provides for TCP/IP.

NI-VISA 3.0 or later allows you to communicate as a controller to PNT devices. NI-VISA is configured to detect USBTMC compliant instruments such as the PNT. To use such a device, plug it in and Windows should detect the new hardware and launch the New Hardware Wizard. Instruct the wizard to search for the driver, which in this case is NI-VISA. If NI-VISA is properly installed, the device will be installed as a USB Test & Measurement Class Device. Open Measurement & Automation Explorer (MAX). The new device will appear in MAX under Device and Interfaces » USB Devices. You can then use this resource name as you would use any GPIB resource.

2.8 USB-TMC Interface Connection and Setup using BNC API

BNC API programming interface supports direct communication to PNT using BNC's proprietary DLL driver libraries. The library allows setup a communication channel though USB, LAN, or GPIB from any programming environment.

Please contact BNC for more detailed documentation, programming samples, and updates on the DLL library.

2.9 GPIB Interface Connection and Setup

2.9.1 General GPIB information

GPIB (General Purpose Interface Bus) is an interface standard for connecting computers and peripherals, which supports the following international standards: IEEE 488.1, IEC-625, IEEE 488.2, and JIS-C1901. The GPIB interface allows you to control the PNT from an external computer. The computer sends commands and instructions to the PNT and receives data sent from the PNT via GPIB. You can connect up to 15 devices in a single GPIB system. By default, the unit is set to GPIB address 1.

The length of cables to connect between devices must be 4 m or less. The total length of connecting cables in a single GPIB system must be 2 m × the number of connected devices (including the controller) or less. You cannot construct the system in which the total cable length exceeds 20 m.

The number of connectors connected to an individual device must be 4 or less. If you connect 5 or more connectors, excessive force is applied to the connector part, which may result in failure.

You can choose the device connection topology from star, linear, and combined. Loop connection is not allowed.

2.10 Using SCPI for PNT

The Standard Commands for Programmable Instrumentation (SCPI) provides a uniform and consistent language to control programmable test and measurement devices in instrumentation systems. The SCPI Standard is built on the foundation of IEEE-488.2, Standard Codes and Formats. It requires conformance to IEEE-488.2, but is pure software standard. SCPI syntax is ASCII text, and therefore can be attached to any computer test language, such as BASIC, C, or C++. It can also be used with Test Application Environments such as LabWindows/CVI, LabVIEW™, or Matlab®. SCPI is hardware independent. SCPI strings can be sent over any instrument interface. It works equally well over USB-TMC, GPIB, or LAN networks.

Please see chapter 5 for detailed description of supported SCPI commands.

3 IEEE-488 Interface Commands

3.1 IEEE Mandated Commands

The required common commands are IEEE-488.2 mandated commands that are defined in the IEEE-488.2 standard and must be implemented by all SCPI compatible instruments. These commands are identified by the asterisk (*) at the beginning of the command keyword. These commands are used to control instrument status registers, status reporting, synchronization, and other common functions.

Commands declared mandatory by *IEEE 488.2*.

*CLS Clear Status Command

*ESE Standard Event Status Enable Command

*ESE? Standard Event Status Enable Query

*ESR? Standard Event Status Register Query

*IDN? Identification Query

*OPC Operation Complete Command

*OPC? Operation Complete Query

*RST Reset Command

*SRE Service Request Enable Command

*SRE? Service Request Enable Query

*STB? Read Status Byte Query

*TST? Self-Test Query

*WAI Wait-to-Continue Command

3.1.1 *CLS

The Clear Status (CLS) command clears the status byte by emptying the error queue and clearing all the event registers including the Data Questionable Event Register, the Standard Event Status Register, the Standard Operation Status Register and any other registers that are summarized in the status byte.

3.1.2 *ESE <data>

The Standard Event Status Enable (ESE) command sets the Standard Event Status Enable Register. The variable <data> represents the sum of the bits that will be enabled.

Range 0–255

Remarks The setting enabled by this command is not affected by instrument preset or *RST. However, cycling the instrument power will reset this register to zero.

3.1.3 *ESE?

The Standard Event Status Enable (ESE) query returns the value of the Standard Event Status Enable Register.

NOTE: Reading the Standard Event Status Register clears it

Remarks The Register is not affected by instrument preset or *RST. However, cycling the instrument power will reset this register to zero.

3.1.4 *IDN?

The Identification (IDN) query outputs an identifying string. The response will show the following information: <company name>, <model number>, <serial number>, <firmware revision>

3.1.5 *OPC

The Operation Complete (OPC) command sets bit 0 in the Standard Event Status Register when all pending operations have finished.

The Operation Complete command causes the device to set the operation complete bit (bit 0) in the Standard Event Status Register when all pending operations have been finished.

3.1.6 *OPC?

The Operation Complete (OPC) query returns the ASCII character 1 in the Standard Event Status Register when all pending operations have finished.

This query stops any new commands from being processed until the current processing is complete. This command blocks the communication until *all* operations are complete (i.e. the timeout setting should be longer than the longest sweep).

3.1.7 *OPT?

The options (OPT) query returns a comma-separated list of all of the options currently installed on the instrument.

3.1.8 *RST

The Reset (RST) command resets most instrument functions to factory- defined conditions.

Remarks Each command shows the [*RST] default value if the setting is affected.

3.1.9 *SRE <data>

The Service Request Enable (SRE) command sets the value of the Service Request Enable Register. The variable <data> is the decimal sum of the bits that will be enabled. Bit 6 (value 64) is ignored and cannot be set by this command.

Range 0–255

The setting enabled by this command is not affected by instrument preset or

*RST. However, cycling the instrument power will reset it to zero.

3.1.10 *SRE?

The Service Request Enable (SRE) query returns the value of the Service Request Enable Register.

Range 0–63 & 128-191

3.1.11 *STB?

The Read Status Byte (STB) query returns the value of the status byte including the master summary status (MSS) bit.

Range 0–255

3.1.12 *TRG

The Trigger (TRG) command triggers the device if LAN is the selected trigger source, otherwise, *TRG is ignored.

3.1.13 *TST?

The Self-Test (TST) query initiates the internal self- test and returns one of the following results:

0 This shows that all tests passed.

1 This shows that one or more tests failed.

3.1.14 *WAI

The Wait- to- Continue (WAI) command causes the instrument to wait until all pending commands are completed, before executing any other commands.

4 SCPI Commands

This chapter provides an introduction to SCPI programming that includes descriptions of the command types, hierarchical command structure, data parameters, and notational conventions. Information on PNT status system and trigger system programming is also provided.

4.1 Introduction

Standard Commands for Programmable Instruments (SCPI) is the new instrument command language for controlling instruments that goes beyond *IEEE 488.2* to address a wide variety of instrument functions in a standard manner. SCPI promotes consistency, from the remote programming standpoint, between instruments of the same class and between instruments with the same functional capability. For a given measurement function such as frequency or voltage, SCPI defines the specific command set that is available for that function. Thus, two oscilloscopes made by different manufacturers could be used to make frequency measurements in the same way. It is also possible for a SCPI counter to make a frequency measurement using the same commands as an oscilloscope. SCPI commands are easy to learn, self-explanatory and account for both novice and expert programmer's usage. Once familiar with the organization and structure of SCPI, considerable efficiency gains can be achieved during control program development, independent of the control program language selected.

A key to consistent programming is the reduction of multiple ways to control similar instrument functions. The philosophy of SCPI is for the same instrument functions to be controlled by the same SCPI commands. To simplify learning, SCPI uses industry-standard names and terms that are manufacturer and customer supported.

The advantage of SCPI for the ATE system programmer is reducing the time learning how to program new SCPI instruments after programming their first SCPI instrument.

Programmers who use programming languages such as BASIC, C, FORTRAN, etc., to send instrument commands to instruments will benefit from SCPI. Also, programmers who implement instrument device drivers for ATE program generators and/or software instrument front panels will benefit by SCPI's advantages. SCPI defines instrument commands, parameters, data, and status. It is not an application package, programming language or software intended for instrument front panel control.

SCPI is designed to be layered on top of the hardware-independent portion of *IEEE 488.2.*

4.2 SCPI Command Types

SCPI commands, which are also referred to as SCPI instructions, are messages to the instrument to perform specific tasks. The PNT command set includes:

- “Common” commands (IEE488.2 mandated commands)
- SCPI required commands
- SCPI optional commands (per SCPI 1999.0)
- SCPI compliant commands that are unique to the PNT. Not all of the commands supported by the instrument are taken from the SCPI standard; however, their syntax follows SCPI rules.

4.3 SCPI Command Syntax

Typical SCPI commands consist of one or more keywords, parameters, and punctuation. SCPI command keywords are not case-sensitive. Lowercase and uppercase letters are considered equivalent.

Except for common commands, each keyword has a long and a short form. In this manual, the long form is presented with the short form in uppercase and the remainder in lowercase. Unrecognized versions of long form or short form commands, or improper syntax, will generate an error.

For example, the following commands are equivalent:

```
:SENSe:FREQuency:EXECute
:SENS:FREQ:EXEC
:sens:freq:exec
```

Structure of a Command Line

A command line may consist of one or several commands. If the command line is transferred via GPIB, it is terminated by an EOI together with the last data byte. If the command line is transferred via LAN, it is terminated by a LF character (0x0A).

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. A colon ":" at the beginning of a command marks the root node of the command tree.

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.

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.

If the response to a query consists of a single value, it is transferred as ASCII text, terminated by a LF character (0x0A).

If the response consists of a list of values, it is typically transferred as binary block data. Block data is formatted according to IEEE 488.2. See section 7 for more details.

Example 1: ASCII command with ASCII response

```
CALC:FREQ?              //Query to calculate and transfer frequency
5000000000.000          //Response in Hz transferred as ASCII text
```

Example 2: ASCII command with block data response

```
CALC:PN:TRAC:FREQ?      //Query to transfer list of offset frequencies
[block data]            //Response formatted as binary block data
```

Parameters

Most commands require a parameter to be specified. The parameters must be separated from the header by a "white space". Permissible parameters are numerical values, Boolean parameters, text, character strings and block data. The type of parameter required for the respective command and the permissible range of values are specified in the command description.

Numerical values Numerical 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 values must be in the value range –9.9E37 to 9.9E37. The exponent is introduced by an "E" or "e". Entry of the exponent alone is not allowed.

Units In the case of physical quantities, the unit can be entered. Permissible unit prefixes are G (giga), MA (mega, MHZ are also permissible), K (kilo), M (milli), U (micro) and N (nano). If the unit is missing, the basic unit is used.

Boolean Parameters Boolean parameters represent two states. The ON state (logically true) is represented by ON or a numerical value unequal to 0. The OFF state (logically false) is represented by OFF or the numerical value 0. Querying a boolean parameter returns the numerical value 0 or 1.

4.4 Hierarchical Command Structure

All SCPI commands, except the common commands, are organized in a hierarchical structure similar to the inverted tree file structure used in most computers. The SCPI standard refers to this structure as “the Command Tree.” The command keywords that correspond to the major instrument control

functions are located at the top of the command tree. The command keywords for the PNT SCPI command set are shown below.

:ABORt

:CALCulate

The purpose of the CALCulate block is to convert or derive sensed data into a form more useful to the application. Typical calculations include converting units, and postprocessing calculations (for example, calculation of jitter of a phase noise trace). The CALCulate commands are described in the CALCulate subsystem.

:DIAGnostic

:INPut

The purpose of the INPut block is to condition the incoming signal before it is converted into data by the SENSe block. INPut block functions include filtering, biasing, frequency conversion (such as a mixer or prescaler function), and attenuation. The INPut block appears in the SCPI tree under the INPut subsystem. The implementation of this subsystem is optional for those instruments that have no INPut block characteristics.

:INITiate

:SENSe

The purpose of the SENSe block is to convert signal(s) into internal data that can be manipulated by normal computer techniques. The commands associated with the SENSe block control the various characteristics of the conversion process. Examples are range, resolution, gate time, normal mode rejection, etc. This block does not include any mathematical manipulation of the data after it has been converted

:STATus

:SYSTem

:TRIGger

The purpose of the TRIGger block is to provide an instrument with synchronization capability with external events. The TRIGger block appears in the SCPI tree as TRIGger, ARM, INITiate, and ABORt subsystems.

:UNIT

All PNT SCPI commands, except the :ABORt command, have one or more subcommands (keywords) associated with them to further define the instrument function to be controlled. The subcommand keywords may also have one or more associated subcommands (keywords). Each subcommand level adds another layer to the command tree. The command keyword and

its associated subcommand keywords form a portion of the command tree called a command subsystem.

4.5 Status System Programming

The PNT implements the status byte register, the Service Request Enable Register, the Standard Event Status Register, and the Standard Event Status Enable Register.

The PNT status system consists of the following SCPI-defined status reporting structures:

- The Instrument Summary Status Byte
- The Standard Event Status Group
- The Operation Status Group
- The Questionable Status Group

The following paragraphs describe the registers that make up a status group and explain the status information that each status group provides.

4.6 Status Registers

In general, a status group consists of a condition register, a transition filter, an event register, and an enable register. Each component is briefly described in the following paragraphs.

Condition Register

The condition register is continuously updated to reflect the current status of the PNT. There is no latching or buffering for this register, it is updated in real time. Reading the contents of a condition register does not change its contents.

Transition Filter

The transition filter is a special register that specifies which types of bit state changes in the condition register will set corresponding bits in the event register. Negative transition filters (NTR) are used to detect condition changes from True (1) to False (0); positive transition filters (PTR) are used to detect condition changes from False (0) to True (1). Setting both positive and negative filters True allows an event to be reported anytime the condition changes. Transition filters are read-write. Transition filters are unaffected by queries or *CLS (clear status) and *RST commands. The command :STATus:PRESet sets all negative transition filters to all 0's and sets all positive transition filters to all 1's.

Event Register

The event register latches transition events from the condition register as specified by the transition filter. Bits in the event register are latched, and once set they remain set until cleared by a query or a *CLS command Event registers are read only.

Enable Register

The enable register specifies the bits in the event register that can produce a summary bit. The PNT logically ANDs corresponding bits in the event and enable registers, and ORs all the resulting bits to obtain a summary bit. Summary bits are recorded in the Summary Status Byte. Enable registers are read-write. Querying an enable register does not affect it. The command :STATus:PRESet sets the Operation Status Enable register and the Questionable Status Enable register to all 0's.

4.7 Status Group Reporting

The state of certain PNT hardware and operational events and conditions can be determined by programming the status system. Three lower status groups provide status information to the Summary Status Byte group. The Summary Status Byte group is used to determine the general nature of an event or condition and the other status groups are used to determine the specific nature of the event or condition.

Summary Status Byte Group

The Summary Status Byte group, consisting of the Summary Status Byte Enable register and the Summary Status Byte, is used to determine the general nature of an PNT event or condition. The bits in the Summary Status Byte provide the following:

4.7.1 Standard Event Status Group

The Standard Event Status group, consisting of the *Standard Event Status register* (an Event register) and the *Standard Event Status Enable register*, is used to determine the specific event that set bit 5 of the Summary Status Byte.

The bits in the *Standard Event Status register* provide the following:

Bit	Description
0	Set to indicate that all pending PNT operations were completed following execution of the "*OPC" command.
1	Request control
2	Set to indicate that a query error has occurred. Query errors have SCPI error codes from –499 to –400.
3	Set to indicate that a device-dependent error has occurred. Device-dependent errors have SCPI error codes from –399 to –300 and 1 to 32767.

4 Set to indicate that an execution error has occurred. Execution errors have SCPI error codes from –299 to –200.

5 Set to indicate that a command error has occurred. Command errors have SCPI error codes from –199 to –100.

6 User request

7 Power on

Standard Event Status Enable register (ESE commands)

4.7.2 Operation Status Group

The Operation Status group, consisting of the Operation Condition register, the Operation Positive Transition register, the Operation Negative Transition register, the Operation Event register, and the Operation Event Enable register, is used to determine the specific condition that set bit 7 in the Summary Status Byte.

4.7.3 Questionable Status Group

The Questionable Status group, consisting of the Questionable Condition register, the Questionable Positive Transition register, the Questionable Negative Transition register, the Questionable Event register, and the Questionable Event Enable register, is used to determine the specific condition that set bit 3 in the Summary Status Byte.

5 SCPI Command Description

5.1 :ABORt Subsystem

The :ABORt command is a single command subsystem. There are no subcommands or associated data parameters, as shown below. The :ABORt command, along with the :TRIGger and :INITiate commands, comprise the Trigger group of commands.

Command	Parameters	Unit	Remark
:ABORt			

:ABORt

`:ABORt`

Aborts measurements or sweeps in progress. Even if INIT:CONT[:ALL] is set to ON, the measurement will not immediately re-initiate.

5.2 :CALCulate Subsystem

The CALCulate subsystem performs post acquisition data processing. Functions in the SENSe subsystem are related to data acquisition, while the CALCulate subsystem operates on the data acquired by a SENSe function.

Command	Parameters	Unit	Remark
`:CALCulate:FREQuency?`		Hz	
`:CALCulate:POWer?`		dBm	
`:CALCulate:WAIT:AVERage`	NEXT | ALL | <value>[,<value>]		

:CALCulate:FREQuency

`:CALCulate:FREQuency?`

Returns the detected DUT frequency.

***RST** 100000000.000000000

Remarks To be used in combination with `SENS:FREQ:EXEC`, which starts the frequency search. The frequency can be read out with this command after the search.

:CALCulate:POWer

`:CALCulate:POWer?`

Returns the detected DUT power level.

***RST** 0.000000000

Remarks To be used in combination with SENS:FREQ:EXEC, which starts the frequency search. The power level can be read out with this command after the search.

:CALCulate:WAIT:AVERage

`:CALCulate:WAIT:AVERage NEXT|ALL|<value>[,<value>]`

This command requests a preliminary result during the measurement and blocks until the result is ready. The first parameter (required) specifies the target iteration to be saved. `NEXT` specifies the next possible iteration, `ALL` specifies the last iteration of the measurement (i.e. waits for the measurement to finish), an integer as first parameter specifies the target iteration.

The second parameter (optional) defines a timeout in milliseconds. If the command terminates without genearting a preliminary result (because of a timeout or a failed measurement), it will produce an error code. The error code can be queried with `SYST:ERR?` or `SYST:ERR:ALL?`.

5.2.1 :CALCulate:PN branch

Command (PN)	Parameters	Unit	Remark
:CALCulate:PN:TRACe:FREQuency?		Hz	block data (1)
:CALCulate:PN:TRACe:NOISe?		dBc/Hz	block data (1)
:CALCulate:PN:TRACe:SPOT? <value>		dBc/Hz	
:CALCulate:PN:TRACe:SPURious:FREQuency?		Hz	block data (1)
:CALCulate:PN:TRACe:SPURious:POWer?		dBc	block data (1)
:CALCulate:PN:TRACe:FUNCtion:JITTer?		s	
:CALCulate:PN:TRACe:FUNCtion:INTegral?		dBc	
:CALCulate:PN:TRACe:FUNCtion:AVARiance			
:CALCulate:PN:TRACe:FUNCtion:AVARiance:SIGMa?			block data (1)
:CALCulate:PN:TRACe:FUNCtion:AVARiance:TAU?			block data (1)
:CALCulate:PN:PRELiminary:AVERage?			
:CALCulate:PN:PRELiminary:CORRelation?			
:CALCulate:PN:TEST?			

(1) Block data according to IEEE 488.2. See section 7 for more details

:CALCulate:PN:TRACe:FREQuency?

`:CALCulate:PN:TRACe:FREQuency?`

Returns a list of offset frequency points of the most recent measurement in [Hz] as block data.

***RST** empty list

:CALCulate:PN:TRACe:NOISe?

`:CALCulate:PN:TRACe:NOISe?`

Returns a list of phase noise measurements in [dBc/Hz].The measurement points correspond to the frequency points returned by `:CALCulate:PN:TRACe:FREQuency?.`

:CALCulate:PN:TRACe:SPOT?

`:CALCulate:PN:TRACe:SPOT? <val>`

Returns the phase noise value of the last measurement at the parameter defined spot frequency. The parameter is given as offset frequency in [Hz].

***RST** -1000.000000000

:CALCulate:PN:TRACe:SPURious:FREQuency?

`:CALCulate:PN:TRACe:SPURious:FREQuency?`

Returns a list of offset frequencies in [Hz] of the spurious in the active trace as block data.

***RST** empty list

Remarks The spurs are ordered by increasing offset frequencies. The ordering corresponds with the ordering of the power value list returned by `:CALCulate:PN:TRACe:SPURious:POWer?.`

:CALCulate:PN:TRACe:SPURious:POWer?

`:CALCulate:PN:TRACe:SPURious:POWer?`

Returns a list of power values in [dBc] of the spurious in the active trace as block data.

***RST** empty list

Remarks The spurs are ordered by increasing offset frequencies. The ordering corresponds with the ordering of the offset frequency value list returned by `:CALCulate:PN:TRACe:SPURious:FREQuency?.`

:CALCulate:PN:TRACe:FUNCtion:JITTer?

`:CALCulate:PN:TRACe:FUNCtion:JITTer?`

Returns the RMS Jitter of the current trace in [s].

***RST** -1.000000E+00

Remarks The offset frequency range used to calculate the jitter from the phase noise trace, is defined with the command `SENSe:PN:FUNCtion:RANGe <min>,<max>`.

:CALCulate:PN:TRACe:FUNCtion:INTegral?

`:CALCulate:PN:TRACe:FUNCtion:INTegral?`

Returns the integral noise of the current trace in [dBc].

***RST** -1.000000000

Remarks The offset frequency range used to calculate the integral, is defined with the command `SENSe:PN:FUNCtion:RANGe <min>,<max>`.

:CALCulate:PN:TRACe:FUNCtion:AVARiance

`:CALCulate:PN:TRACe:FUNCtion:AVARiance`

Calculates the Allan variance for the current trace.

***RST** empty list

Remarks The offset frequency range used to calculate the Allan variance, is defined with the command `SENSe:PN:FUNCtion:RANGe <min>,<max>`.

:CALCulate:PN:TRACe:FUNCtion:AVARiance:TAU?

`:CALCulate:PN:TRACe:FUNCtion:AVARiance:TAU?`

Returns a list of tau values of the Allan variance for the current trace in [s] as block data.

***RST** empty list

Remarks The offset frequency range used to calculate the Allan variance, is defined with the command `SENSe:PN:FUNCtion:RANGe <min>,<max>`. The Allan variance must be calculated before reading the values using the command `:CALCulate:PN:TRACe:FUNCtion:AVARiance.`

:CALCulate:PN:TRACe:FUNCtion:AVARiance:SIGMa?

`:CALCulate:PN:TRACe:FUNCtion:AVARiance:SIGMa?`

Returns a list of sigma values of the Allan variance for the current trace as block data.

***RST** empty list

Remarks The offset frequency range used to calculate the Allan variance, is defined with the command `:SENSe:PN:FUNCtion:RANGe <min>,<max>`. The

Allan variance must be calculated before reading the values using the command `:CALCulate:PN:TRACe:FUNCtion:AVARiance`.

:CALCulate:PN:PRELiminary:AVERage?

`:CALCulate:PN:PRELiminary:AVERage?`

Returns the number of averages for the current data saved on the device.

***RST** 0

:CALCulate:PN:PRELiminary:CORRelation?

`:CALCulate:PN:PRELiminary:CORRelation?`

Returns the number of correlations for the current data saved on the device.

***RST** 0

:CALCulate:PN:TEST?

`:CALCulate:PN:TEST?`

Returns a comma separated list of values, defined by the test set command `SENSe:PN:TEST <definition>`.

***RST** empty list

Remarks The elements of the test set are stored during a measurement that is performed after the test set has been defined.

5.2.2 :CALCulate:VCO branch

Command (VCO Testing)	Parameters	Unit	Remark
:CALCulate:VCO:TRACe:FREQuency?		Hz	block data (1)
:CALCulate:VCO:TRACe:ISupply?		A	block data (1)
:CALCulate:VCO:TRACe:KPUSh?		Hz / V	block data (1)
:CALCulate:VCO:TRACe:KVCO?		Hz / V	block data (1)
:CALCulate:VCO:TRACe:PNoise?	{1 | 2 | 3 | 4}	dBc/Hz	block data (1)
:CALCulate:VCO:TRACe:POWer?		dBm	block data (1)
:CALCulate:VCO:TRACe:VOLTage?		V	block data (1)
:CALCulate:VCO:ITERation?			
:CALCulate:VCO:WAIT	{NEXT | ALL}, {<value>}		

(1) Block data according to IEEE 488.2. See section 7 for more details

:CALCulate:VCO:TRACe:FREQuency?

`:CALCulate:VCO:TRACe:FREQuency?`

Returns a list of frequency values in [Hz] measured at each tune voltage point of the current measurement as block data.

***RST** empty list

:CALCulate:VCO:TRACe:ISUPply?

`:CALCulate:VCO:TRACe:ISUPply?`

Returns a list of supply current values in [A] measured at each tune voltage point of the current measurement as block data.

***RST** empty list

:CALCulate:VCO:TRACe:KPUSh?

`:CALCulate:VCO:TRACe:KPUSh?`

Returns a list of pushing values in [Hz/V] measured at each tune voltage point of the current measurement as block data.

***RST** empty list

:CALCulate:VCO:TRACe:KVCO?

`:CALCulate:VCO:TRACe:KVCO?`

Returns a list of Kv values in [Hz/V] measured at each tune voltage point of the current measurement as block data.

***RST** empty list

:CALCulate:VCO:TRACe:PNoise?

`:CALCulate:VCO:TRACe:PNoise? 1|2|3|4`

Returns a list of phase noise values in [dBc/Hz] measured at each tune voltage point of the current measurement as block data. The parameter 1-4 selects the offset frequency from the set defined by the :SENS:VCO:TEST:PNoise:OFFSet command.

***RST** empty list

:CALCulate:VCO:TRACe:POWer?

`:CALCulate:VCO:TRACe:POWer?`

Returns a list of power values in [dBm] measured at each tune voltage point of the current measurement as block data.

***RST** empty list

:CALCulate:VCO:TRACe:VOLTage?

`:CALCulate:VCO:TRACe:POWer?`

Returns a list of tune voltage values in [V] measured at each tune voltage point of the current measurement as block data.

***RST** empty list

:CALCulate:VCO:ITERation?

`:CALCulate:VCO:ITERation?`

Returns the iteration of the current VCO Characterization measurement.

***RST** 0

Remarks `CALC:VCO:WAIT` must have been called before using this command.

:CALCulate:VCO:WAIT

`:CALCulate:VCO:WAIT NEXT|ALL|<value>`

This command requests a preliminary result during the measurement and blocks until the result is ready. The first parameter (required) specifies the target iteration to be saved. `NEXT` specifies the next possible iteration, `ALL` specifies the last iteration of the measurement (i.e. waits for the measurement to finish) and an integer specifies the specific iteration requested.

The second parameter (optional) defines a timeout in miliseconds. If the command terminates without generating a preliminary result, it will produce an error. This error can be queried with `SYST:ERR?` or `SYST:ERR:ALL?`.

5.2.3 :CALCulate:AN branch

Command (AN)	Parameters	Unit	Remark
:CALCulate:AN:TRACe:FREQuency?		Hz	block data (1)
:CALCulate:AN:TRACe:NOISe?		dBc/Hz	block data (1)
:CALCulate:AN:TRACe:SPOT? <value>		dBc/Hz	
:CALCulate:AN:TRACe:SPURious:FREQuency?		Hz	block data (1)
:CALCulate:AN:TRACe:SPURious:POWer?		dBc	block data (1)

:CALCulate:AN:PREL:AVERage?			
:CALCulate:AN:PRELiminary:CORRelation?			

(1) Block data according to IEEE 488.2. See section 7 for more details

:CALCulate:AN:TRACe:FREQuency?

`:CALCulate:AN:TRACe:FREQuency?`

Returns a list of offset frequency values in [Hz] of the current measurement as block data.

***RST** empty list

:CALCulate:AN:TRACe:NOISe?

`:CALCulate:AN:TRACe:NOISe?`

Returns a list of amplitude noise spectrum values in [dBc/Hz] of the current measurement as block data.

***RST** empty list

:CALCulate:AN:TRACe:SPOT?

`:CALCulate:AN:TRACe:SPOT? <val>`

Returns the spot noise value at the specified spot noise offset frequency in [dBc/Hz]. The parameter defines the spot noise offset frequency in [Hz].

***RST** -1000.000000000

:CALCulate:AN:TRACe:SPURious:FREQuency?

`:CALCulate:AN:TRACe:SPURious:FREQuency?`

Returns a list of offset frequencies in [Hz] of the spurs in the active trace as block data.

***RST** empty list

:CALCulate:AN:TRACe:SPURious:POWer?

`:CALCulate:AN:TRACe:SPURious:POWer?`

Returns a list of power values in [dBc] of the spurs in the active trace as block data.

***RST** empty list

:CALCulate:AN:PRELiminary:AVERage?

`:CALCulate:AN:PRELiminary:AVERage?`

Returns the number of averages of the current preliminary result.

***RST** 0

:CALCulate:AN:PRELiminary:CORRelation?

`:CALCulate:AN:PRELiminary:CORRelation?`

Returns the number of correlations of the current preliminary result.

***RST** 0

5.2.4 :CALCulate:FN branch

Command (FN)	Parameters	Unit	Remark
:CALCulate:FN:TRACe:FREQuency?		Hz	block data (1)
:CALCulate:FN:TRACe:NOISe?		dBc/Hz	block data (1)
:CALCulate:FN:TRACe:SPOT? <value>		dBc/Hz	
:CALCulate:FN:TRACe:SPURious:FREQuency?		Hz	block data (1)
:CALCulate:FN:TRACe:SPURious:POWer?		dBc	block data (1)
:CALCulate:FN:PREL:AVERage?			
:CALCulate:FN:PRELiminary:CORRelation?			

(1) Block data according to IEEE 488.2. See section 7 for more details

:CALCulate:FN:TRACe:FREQuency?

`:CALCulate:FN:TRACe:FREQuency?`

Returns a list of offset frequency values in [Hz] of the current measurement as block data.

***RST** empty list

:CALCulate:FN:TRACe:NOISe?

`:CALCulate:FN:TRACe:NOISe?`

Returns a list of phase noise spectrum values in [dBc/Hz] of the current measurement as block data.

***RST** empty list

:CALCulate:FN:TRACe:SPOT?

`:CALCulate:FN:TRACe:SPOT? <val>`

Returns the spot noise value at the specified spot noise offset frequency in [dBc/Hz]. The parameter defines the spot noise offset frequency in [Hz].

***RST** -1000.000000000

:CALCulate:FN:TRACe:SPURious:FREQuency?

`:CALCulate:FN:TRACe:SPURious:FREQuency?`

Returns a list of offset frequencies in [Hz] of the spurs in the active trace as block data.

***RST** empty list

:CALCulate:FN:TRACe:SPURious:POWer?

`:CALCulate:FN:TRACe:SPURious:POWer?`

Returns a list of power values in [dBc] of the spurs in the active trace as block data.

***RST** empty list

:CALCulate:FN:PRELiminary:AVERage?

`:CALCulate:FN:PRELiminary:AVERage?`

Returns the number of averages of the current preliminary result.

***RST** 0

:CALCulate:FN:PRELiminary:CORRelation?

`:CALCulate:FN:PRELiminary:CORRelation?`

Returns the number of correlations of the current preliminary result.

***RST** 0

5.3 :INITiate Subsystem

The :INITiate subsystem controls the state of the PNT trigger system. The subsystem commands and parameters are described below. The :INITiate commands, along with the :ABORt and :TRIGger commands, comprise the Trigger Group of commands.

Command	Parameters	Unit	Remark
:INITiate[:IMMediate]			
:INITiate:CONTinuous	{ON|OFF|1|0}	OFF	(1)

(1) Will be available with firmware 0.5.1

:INITiate[:IMMediate]

```
:INITiate[:IMMediate]
```

Sets the trigger to the armed state. This will start the predifined measurement at the next possible instance.

:INITiate:CONTinuous

```
:INITiate:CONTinuous ON|OFF
```

This command continuously rearms the trigger after completion of a triggered measurement.

5.4 :SENSe Subsystem

The SENSe command subsystem directly affects device- specific settings used to make measurements.

Command	Parameters	Default	Remark
:SENSe:FREQuency:EXECute			
:SENSe:POWer:EXECute			
:SENSe:MODE	{PN | VCO | AN | FN | BB | TRAN}	PN	

:SENSe:FREQuency:EXECute

```
:SENSe:FREQuency:EXECute
```

Starts the frequency search. See the CALCulate subsystem on how to read out the measurement results.

:SENSe:POWer:EXECute

```
:SENSe:POWer:EXECute
```

Starts the power measurement. See the CALCulate subsystem on how to read out the measurement results.

:SENSe:MODE

```
:SENSe:MODE PN|AN|FN|BB|TRAN|VCO
:SENSe:MODE?
```

Sets/gets the active measurement mode.

- PN: phase noise measurement

- AN: amplitude noise measurement
- FN: frequency noise measurement (results are converted to phase noise)
- BB: base band measurement (not yet available)
- TRAN: transient analysis (not yet available)
- VCO: voltage controlled oscillator characterization

5.4.1 :SENSe:PN branch

Command (PN)	Parameters	Default	Remark
:SENSe:PN:KPHI	<value>	0 rad / V	
:SENSe:PN:KPHI:AUTO	{ON | OFF | 1 | 0}	ON	
:SENSe:PN:KPHI:DETect	{ALWays | ONCe | NEVer}	ALWays	
:SENSe:PN:LOBandwidth	{0.1 ~ 10k}	10 Hz	
:SENSe:PN:LOBandwidth:AUTO	{ON | OFF | 1 | 0}	ON	
:SENSe:PN:PREAmplifier	{ON | OFF | 1 | 0}	OFF	
:SENSe:PN:REFerences	{LN | NORM | HIGH | EXT}	NORM	
:SENSe:PN:REFerences#:SENSitivity	{1|2},<value>	1 Hz / V	
:SENSe:PN:REFerences:SENSitivity:EXECute			
:SENSe:PN:REFerences:TUNE:MAX	{1|2},{3 ~ 20}	3 V	(1)
:SENSe:PN:REFerences:TUNE:MIN	{1|2},{-5 ~ TUNE MAX}	0 V	(1)
:SENSe:PN:AVERage	{1 ~ 10k}	1	
:SENSe:PN:CORRelation	{1 ~ 10k}	1	
:SENSe:PN:IFGain	<value>	0 dB	
:SENSe:PN:IFGain:AUTO	{ON | OFF | 1 | 0}	ON	
:SENSe:PN:IFGain:DETect	{ALWays | ONCe | NEVer}	ALWays	
:SENSe:PN:FREQuency	<value>	100e6 Hz	
:SENSe:PN:FREQuency:AUTO	{ON|OFF|1|0}	ON	
:SENSe:PN:FREQuency:DETect	{ALWays | NEVer}	ALWays	

Command (PN)	Parameters	Default	Remark
:SENSe:PN:FREQuency:STARt	{ 0.1 | 0.5 | 1 | 10 | 100 | 1k | 10k | 100k }	100 Hz	
:SENSe:PN:FREQuency:STOP	{1k | 10k | 100k | 1M |10M | 50M }	50e6 Hz	
:SENSe:PN:FUNCtion:RANGe	{0.1 ~ 50M},{0.1 ~ 50M}	10,50E6 Hz	
:SENSe:PN:POWer	<value>	0 dBm	(1)
:SENSe:PN:POWer:AUTO	{ON|OFF|1|0}	ON	(1)
:SENSe:PN:POWer:DETect	{ALWays | NEVer}	ON	(1)
:SENSe:PN:PPD	{1 ~ 500}	250	
:SENSe:PN:RESet			
:SENSe:PN:SPURious:OMISsion	{ON | OFF | 1 | 0}	ON	
:SENSe:PN:SMOothing:APERture	{0.05 ~ 20}	0.05 %	
:SENSe:PN:SMOothing:STATe	{ON | OFF | 1 | 0}	OFF	
:SENSe:PN:TEST	Testset definition		

(1) Will be available with firmware 0.5.1

:SENSe:PN:KPHI

```
:SENSe:PN:KPHI <value>
:SENSe:PN:KPHI?
```

Sets/gets Kphi in [rad/V].

***RST** 0.000000000

:SENSe:PN:KPHI:AUTO

```
:SENSe:PN:KPHI:AUTO ON|OFF|1|0
:SENSe:PN:KPHI:AUTO?
```

Enables/disables the automatic Kphi detection at the start of the measurement.

***RST** 1

:SENSe:PN:KPHI:DETect

```
:SENSe:PN:KPHI:DETect ALWays|ONCe|NEVer
:SENSe:PN:KPHI:DETect?
```

Sets how often Kphi detection should be performed. `ALWays`: Perform it for every measurement; `ONCe`: Perform it only if it hasn't been performet yet since startup of the instrument; `NEVer`: Always skip it.

***RST** ALW

:SENSe:PN:LOBandwidth

```
:SENSe:PN:LOBandwidth <value>
:SENSe:PN:LOBandwidth?
```

Sets/gets the PLL bandwidth for the selected channel in [Hz]. The value can be set between 0.1 Hz and 10 kHz.

***RST** 1.000000000

:SENSe:PN:LOBandwidth:AUTO

```
:SENSe:PN:LOBandwidth:AUTO ON|OFF|1|0
:SENSe:PN:LOBandwidth:AUTO?
```

Enables/disables the automatic selection of the optimal bandwidth during the measurement.

***RST** 1

:SENSe:PN:PREAmplifier

```
:SENSe:PN:PREAmplifier ON|OFF|1|0
:SENSe:PN:PREAmplifier?
```

Enables/disables the preamplifier.

***RST** 0

:SENSe:PN:REFerences

```
:SENSe:PN:REFerences LN|NORM|HIGH|EXT
:SENSe:PN:REFerences?
```

Selects the reference used for the measurement. The options include NORM for standard internal references, LN for low noise internal references (only available with option LN), HIGH for high sensitivity internal references and EXT for external references.

***RST** NORM

:SENSe:PN:REFerences#:SENSitivity

```
:SENSe:PN:REFerences#:SENSitivity <value>
:SENSe:PN:REFerences#:SENSitivity?
```

Sets/gets the sensitivity for the specified reference #.

***RST** 10.000000000

:SENSe:PN:REFerences:SENSitivity:EXECute

`:SENSe:PN:REFerences:SENSitivity:EXECute`

Starts the sensitivity measurement of the selected reference (not yet available).

:SENSe:PN:AVERage

`:SENSe:PN:AVERage <val>`
`:SENSe:PN:AVERage?`

Sets/gets the average count of the measurement.

Range <val>: 1 - 10000

***RST** 1

:SENSe:PN:CORRelation

`:SENSe:PN:CORRelation <val>`
`:SENSe:PN:CORRelation?`

Sets/gets the correlation count of the measurement.

Range <val>: 1 - 10000

***RST** 1

:SENSe:PN:IFGain

`:SENSe:PN:IFGain <val>`
`:SENSe:PN:IFGAIN?`

Sets/gets the IF gain for the measurement.

Range <val>: 20-60

***RST** 0

:SENSe:PN:IFGain:AUTO

`:SENSe:PN:IFGain:AUTO ON|OFF|1|0`
`:SENSe:PN:IFGAIN:AUTO?`

Enables/disables the automatic IF gain setting.

***RST** 1

:SENSe:PN:IFGain:DETect

`:SENSe:PN:IFGain:DETect ALWays|ONCe|NEVer`
`:SENSe:PN:IFGAIN:DETect?`

Sets how often the IF gain should be automatically determined. `ALWays`: Perform it for every measurement; `ONCe`: Perform it only if it hasn't been performed yet since startup of the instrument; `NEVer`: Always skip it.

***RST** ALW

:SENSe:PN:FREQuency

```
:SENSe:PN:FREQuency <value>
:SENSe:PN:FREQuency?
```

Sets/gets the DUT frequency in [Hz].

***RST** 100000000.000000000

:SENSe:PN:FREQuency:AUTO

```
:SENSe:PN:FREQuency:AUTO ON|OFF|1|0
:SENSe:PN:FREQuency:AUTO?
```

Enables/disables the automatic frequency search at the start of the measurement.

***RST** 1

:SENSe:PN:FREQuency:DETect

```
:SENSe:PN:FREQuency:DETect ALWays|ONCe|NEVer
:SENSe:PN:FREQuency:DETect?
```

Sets how often the automatic frequency search should be performed (if activated via `SENS:PN:FREQ:AUTO`). `ALWays`: Perform it for every measurement; `ONCe`: Perform it only if it hasn't been performed yet since startup of the instrument; `NEVer`: Always skip it.

***RST** 1

:SENSe:PN:FREQuency:STARt

```
:SENSe:PN:FREQuency:STARt <value>
:SENSe:PN:FREQuency:STARt?
```

Sets/gets the start offset frequency.

***RST** 10.000000000

:SENSe:PN:FREQuency:STOP

```
:SENSe:PN:FREQuency:STOP <value>
:SENSe:PN:FREQuency:STOP?
```

Sets/gets the stop offset frequency.

***RST** 100000000.000000000

:SENSe:PN:FUNCtion:RANGe

```
:SENSe:PN:FUNCtion:RANGe <min>,<max>
```

Sets the frequency offset range for the FUNC subsystem. This system offers statistical analysis of the measurement data.

:SENSe:PN:POWer

```
:SENSe:PN:POWer <value>
:SENSe:PN:POWer?
```

Sets/gets the DUT power in dBm.

***RST** 0.0

:SENSe:PN:POWer:AUTO

```
:SENSe:PN:POWer:AUTO ON|OFF|1|0
:SENSe:PN:POWer:AUTO?
```

Enables/disables the automatic power measurement at the start of the measurement.

***RST** 1

:SENSe:PN:POWer:DETect

```
:SENSe:PN:POWer:DETect ALWays|ONCe|NEVer
:SENSe:PN:POWer:DETect?
```

Sets how often the automatic frequency search should be performed (if activated via SENS:PN:FREQ:AUTO). ALWays: Perform it for every measurement; ONCe: Perform it only if it hasn't been performed yet since startup of the instrument; NEVer: Always skip it.

***RST** ALW

:SENSe:PN:PPD

```
:SENSe:PN:PPD <value>
:SENSe:PN:PPD?
```

Sets/gets the number of points per decade for the trace.

***RST** 120

:SENSe:PN:RESet

```
:SENSe:PN:RESet
```

Resets the measurement. The measurement configuration will remain, but the detect states will be reset (ONCe will be active again).

:SENSe:PN:SPURious:OMISsion

```
:SENSe:PN:SPURious:OMISsion ON|OFF|1|0
:SENSe:PN:SPURious:OMISsion?
```

Enables/disables spur omission for the trace and statistical analysis.

***RST** 1

:SENSe:PN:SMOothing:APERture

```
:SENSe:PN:SMOothing:APERture <value>
:SENSe:PN:SMOothing:APERture?
```

Sets/gets the smoothing aperture of the trace in [%].

***RST** 0.100000000

:SENSe:PN:SMOothing:STATe

```
:SENSe:PN:SMOothing:STATe ON|OFF
:SENSe:PN:SMOothing:STATe?
```

Enables/disables trace smoothing.

***RST** 1

:SENSe:PN:TEST

```
:SENSe:PN:TEST <definition>
:SENSe:PN:TEST?
```

Sets/gets the test set definition. The test set definition is a comma separated list of keywords:

O<val>	phase noise in [dBc/Hz] at offset <val> specified in [Hz]
F	detected frequencyin [Hz]
P	detected power level in [dBm]
J	jitter value in [fs], offset range is defined with the command `SENSe:PN:FUNCtion:RANGe <min>,<max>`
I	integrated phase noise in [dBc] (same offset range as jitter)
D	residual PM in [udeg] (same offset range as jitter)
R	residual PM in [urad] (same offset range as jitter)
M	residual FM in [Hz] (same offset range as jitter)

Example `SENS:PN:TEST O1e3,O1e5,O1e6,O5e6,F,P,J`

This test set contains the phase noise values in [dBc/Hz] at 1kHz, 100kHz, 1MHz, 5MHz, the detected frequency in [Hz], power level in [dBm] and jitter in [fs].

5.4.2 :SENSe:VCO branch

Command (VCO Tesing)	Parameters	Default	Remark
:SENSe:VCO:TEST:FREQuency	{ON | OFF | 1 | 0}	ON	
:SENSe:VCO:TEST:ISUPply	{ON | OFF | 1 | 0}	ON	
:SENSe:VCO:TEST:KPUShing	{ON | OFF | 1 | 0}	ON	
:SENSe:VCO:TEST:KVCO	{ON | OFF | 1 | 0}	ON	
:SENSe:VCO:TEST:PNoise	{ON | OFF | 1 | 0}	OFF	
:SENSe:VCO:TEST:PNoise:OFFSet | :SENSe:VCO:TEST:PNoise:OFFSet#?	{10 ~ 40M}, {10 ~ 40M}, {10 ~ 40M}, {10 ~ 40M} | {1 | 2 | 3 | 4}	10 kHz, 100 kHz, 1 MHz, 10 MHz	
:SENSe:VCO:TEST:PNoise:COUNt?		0	
:SENSe:VCO:TEST:POWer	{ON | OFF | 1 | 0}	ON	
:SENSe:VCO:TYPE	{VCO | VCXO}	VCO	
:SENSe:VCO: VOLTage:POINts	{1 ~ 1000}	10	
:SENSe:VCO:VOLTage:STARt	{ -5.0 ~ 21.0 V}	0.0 V	
:SENSe:VCO: VOLTage:STOP	{ -5.0 ~ 21.0 V}	5.0 V	

:SENSe:VCO:TEST:FREQuency

```
:SENSe:VCO:TEST:FREQuency ON|OFF|1|0
:SENSe:VCO:TEST:FREQuency?
```

Enables/disables the frequency parameter for the measurement.

***RST** 1

:SENSe:VCO:TEST:ISUPply

```
:SENSe:VCO:TEST:ISUPply ON|OFF|1|0
:SENSe:VCO:TEST:ISUPply?
```

Enables/disables the acquisition of current at the supply port for the measurement.

***RST** 1

:SENSe:VCO:TEST:KPUShing

```
:SENSe:VCO:TEST:KPUShing ON|OFF|1|0
:SENSe:VCO:TEST:KPUShin?
```

Enables/disables the pushing parameter for the measurement.

***RST** 0

:SENSe:VCO:TEST:KVCO

```
:SENSe:VCO:TEST:KVCO ON|OFF|1|0
:SENSe:VCO:TEST:KVCO?
```

Enables/disables the tune sensitivity parameter for the measurement.

***RST** 1

:SENSe:VCO:TEST:PNoise

```
:SENSe:VCO:TEST:PNoise ON|OFF|1|0
:SENSe:VCO:TEST:PNoise?
```

Enables/disables the phase noise parameter for the measurement.

***RST** 1

:SENSe:VCO:TEST:PNoise:OFFSet

```
:SENSe:VCO:TEST:PNoise:OFFSet <val1>[,<val2>][,<val3>][,<val4>]
:SENSe:VCO:TEST:PNoise:OFFSet#?
```

This command sets up to 4 offset frequencies at which the phase noise is measured. At least 1 parameter is required. Blank parameters are set to 0 (disabled).

The query returns the set frequency for the specified offset #.

***RST** 1: 10000.000000000

2: 100000.000000000

3: 1000000.000000000

4: 10000000.000000000

:SENSe:VCO:TEST:PNoise:COUNt

```
:SENSe:VCO:TEST:COUNt?
```

Returns the number of offset frequencies that are set.

***RST** 4

:SENSe:VCO:TEST:POWer

```
:SENSe:VCO:TEST:POWer ON|OFF|1|0
:SENSe:VCO:TEST:POWer?
```

Enables/disables the power parameter during the measurement.

***RST** 1

:SENSe:VCO:TYPE

```
:SENSe:VCO:TYPE VCO|VCXO
:SENSe:VCO:TYPE?
```

Select the DUT type for the measurement. Distinguish between slow (VCX0) and fast (VCO) tuning sensitivities.

***RST** VCO

:SENSe:VCO:VOLTage:POINts

```
:SENSe:VCO:VOLTage:POINts <value>
:SENSe:VCO:VOLTage:POINTs?
```

Sets/gets the number of voltage points to use in the measurement.

***RST** 10

:SENSe:VCO:VOLTage:STARt

```
:SENSe:VCO:VOLTage:STARt <value>
:SENSe:VCO:VOLTage:STARt?
```

Sets/gets the start tuning voltage for the measurement.

***RST** 0

:SENSe:VCO:VOLTage:STOP

```
:SENSe:VCO:VOLTage:STOP <value>
:SENSe:VCO:VOLTage:STOP?
```

Sets/gets the stop tuning voltage for the measurement.

***RST** 5

5.4.3 :SENSe:AN branch

Command (AN)	Parameters	Default	Remark
:SENSe:AN:AVERage	{1 ~ 10k}	1	
:SENSe:AN:CORRelation	{1 ~ 10k}	1	
:SENSe:AN:FREQuency	<value>	100e6 Hz	
:SENSe:AN:FREQuency:AUTO	{ON|OFF|1|0}	ON	
:SENSe:AN:FREQuency:DETect	{ALWays | NEVer}	ALWays	
:SENSe:AN:FREQuency:STARt	{ 0.1 | 0.5 | 1 | 10 | 100 | 1k | 10k | 100k }	100 Hz	
:SENSe:AN:FREQuency:STOP	{1k | 10k | 100k | 1M |10M | 40M }	40e6 Hz	
:SENSe:AN:PPD	{1 ~ 500}	250	
:SENSe:AN:RESet			
:SENSe:AN:SPURious:OMISsion	{ON | OFF | 1 | 0}	ON	
:SENSe:AN:SPURious:THReshold	{1 ~ 70}	10 dB	
:SENSe:AN:SMOothing:APERture	{0.05 ~ 20}	0.05 %	
:SENSe:AN:SMOothing:STATe	{ON | OFF | 1 | 0}	OFF	

:SENSe:AN:AVERage

```
:SENSe:AN:AVERage <value>
:SENSe:AN:AVERage?
```

Sets/gets the average count of the measurement.

***RST** 1

:SENSe:AN:CORRelation

```
:SENSe:AN:CORRelation <value>
:SENSe:AN:CORRelation?
```

Sets/gets the correlation count of the measurement.

***RST** 1

:SENSe:AN:FREQuency

```
:SENSe:AN:FREQuency <value>
:SENSe:AN:FREQuency?
```

Sets/gets the DUT frequency in [Hz].

***RST** 100000000.000000000

:SENSe:AN:FREQuency:AUTO

```
:SENSe:AN:FREQuency:AUTO ON|OFF|1|0
:SENSe:AN:FREQuency:AUTO?
```

Enables/disables the automatic frequency search at the start of the measurement.

***RST** 1

:SENSe:AN:FREQuency:DETect

```
:SENSe:AN:FREQuency:DETect ALWays|ONCe|NEVer
:SENSe:AN:FREQuency:DETect?
```

Sets how often the automatic frequency search should be performed (if activated via `SENS:AN:FREQ:AUTO`). `ALWays`: Perform it for every measurement; `ONCe`: Perform it only if it hasn't been performed yet since startup of the instrument; `NEVer`: Always skip it.

***RST** ALW

:SENSe:AN:FREQuency:STARt

```
:SENSe:AN:FREQuency:STARt <value>
:SENSe:AN:FREQuency:STARt?
```

Sets/gets the start offset frequency for the measurement.

***RST** 10.000000000

:SENSe:AN:FREQuency:STOP

```
:SENSe:AN:FREQuency:STOP <value>
:SENSe:AN:FREQuency:STOP?
```

Sets/gets the stop offset frequency for the measurement.

***RST** 40000000.000000000

:SENSe:AN:PPD

```
:SENSe:AN:PPD <value>
:SENSe:AN:PPD?
```

Sets/gets the number of points per decade for the trace.

***RST** 250

:SENSe:AN:RESet

```
:SENSe:AN:FREQuency:RESet
```

Resets the measurement. The measurement configuration will remain, but the detect states will be reset (`ONCe` will be active again).

:SENSe:AN:SPURious:OMISsion

```
:SENSe:AN:SPURious:OMISsion ON|OFF|1|0
:SENSe:AN:SPURious:OMISsion?
```

Enables/disables spur omission for the trace and statistical analysis.

***RST** 1

:SENSe:AN:SPURious:THREshold

```
:SENSe:AN:SPURious:THREshold <value>
:SENSe:AN:SPURious:THREshold?
```

Sets the threshold for the spur omission in [dB]. If spur omission is `OFF`, spurs that fall under this threshold are still not included.

:SENSe:AN:SMOothing:APERture

```
:SENSe:AN:SMOothing:APERture <value>
:SENSe:AN:SMOothing:APERture?
```

Sets/gets the smoothing aperture of the trace in [%].

***RST** 0.050000000

:SENSe:AN:SMOothing:STATe

```
:SENSe:AN:SMOothing:STATe ON|OFF|1|0
:SENSe:AN:SMOothing:STATe?
```

Enables/disables trace smoothing.

***RST** 0

5.4.4 :SENSe:FN branch

Command (FN)	Parameters	Default	Remark
:SENSe:FN:AVERage	{1 ~ 10k}	1	
:SENSe:FN:CORRelation	{1 ~ 10k}	1	
:SENSe:FN:FREQuency	<value>	100e6 Hz	
:SENSe:FN:FREQuency:AUTO	{ON|OFF|1|0}	ON	
:SENSe:FN:FREQuency:DETect	{ALWays | NEVer}	ALWays	
:SENSe:FN:FREQuency:STARt	{ 0.1 | 0.5 | 1 | 10 | 100 | 1k | 10k | 100k }	100 Hz	
:SENSe:FN:FREQuency:STOP	{1k | 10k | 100k | 1M |10M | 40M }	40e6 Hz	
:SENSe:FN:PPD	{1 ~ 500}	250	
:SENSe:FN:RESet			
:SENSe:FN:SPURious:OMISsion	{ON | OFF | 1 | 0}	ON	
:SENSe:FN:SPURious:THReshold	{1 ~ 70}	10 dB	
:SENSe:FN:SMOothing:APERture	{0.05 ~ 20}	0.05 %	
:SENSe:FN:SMOothing:STATe	{ON | OFF | 1 | 0}	OFF	

:SENSe:FN:AVERage

```
:SENSe:FN:AVERage <value>
:SENSe:FN:AVERage?
```

Sets/gets the average count of the measurement.

***RST** 1

:SENSe:FN:CORRelation

```
:SENSe:FN:CORRelation <value>
:SENSe:FN:CORRelation?
```

Sets/gets the correlation count of the measurement.

***RST** 1

:SENSe:FN:FREQuency

```
:SENSe:FN:FREQuency <value>
:SENSe:FN:FREQuency?
```

Sets/gets the DUT frequency in [Hz].

***RST** 100000000.000000000

:SENSe:FN:FREQuency:AUTO

```
:SENSe:FN:FREQuency:AUTO ON|OFF|1|0
:SENSe:FN:FREQuency:AUTO?
```

Enables/disables the automatic frequency search at the start of the measurement.

***RST** 1

:SENSe:FN:FREQuency:DETect

```
:SENSe:FN:FREQuency:DETect ALWays|ONCe|NEVer
:SENSe:FN:FREQuency:DETect?
```

Sets how often the automatic frequency search should be performed (if activated via `SENS:FN:FREQ:AUTO`). `ALWays`: Perform it for every measurement; `ONCe`: Perform it only if it hasn't been performed yet since startup of the instrument; `NEVer`: Always skip it.

***RST** ALW

:SENSe:FN:FREQuency:STARt

```
:SENSe:FN:FREQuency:STARt <value>
:SENSe:FN:FREQuency:STARt?
```

Sets/gets the start offset frequency for the measurement.

***RST** 10.000000000

:SENSe:FN:FREQuency:STOP

```
:SENSe:FN:FREQuency:STOP <value>
:SENSe:FN:FREQuency:STOP?
```

Sets/gets the stop offset frequency for the measurement.

***RST** 40000000.000000000

:SENSe:FN:PPD

```
:SENSe:FN:PPD <value>
:SENSe:FN:PPD?
```

Sets/gets the number of points per decade for the trace.

***RST** 250

:SENSe:FN:RESet

```
:SENSe:FN:FREQuency:RESet
```

Resets the measurement. The measurement configuration will remain, but the detect states will be reset (`ONCe` will be active again).

:SENSe:FN:SPURious:OMISsion

```
:SENSe:FN:SPURious:OMISsion ON|OFF|1|0
:SENSe:FN:SPURious:OMISsion?
```

Enables/disables spur omission for the trace and statistical analysis.

***RST** 1

:SENSe:FN:SPURious:THREshold

```
:SENSe:FN:SPURious:THREshold <value>
:SENSe:FN:SPURious:THREshold?
```

Sets the threshold for the spur omission in [dB]. If spur omission is `OFF`, spurs that fall under this threshold are still not included.

:SENSe:FN:SMOothing:APERture

```
:SENSe:FN:SMOothing:APERture <value>
:SENSe:FN:SMOothing:APERture?
```

Sets/gets the smoothing aperture of the trace in [%].

***RST** 0.050000000

:SENSe:FN:SMOothing:STATe

```
:SENSe:FN:SMOothing:STATe ON|OFF|1|0
:SENSe:FN:SMOothing:STATe?
```

Enables/disables trace smoothing.

***RST** 0

5.5 :SOURce Subsystem

Command	Parameters	Default	Remark
:SOURce:TUNE:DUT:VOLT	<value>	0	
:SOURce:TUNE:DUT:STAT	{ON|OFF|1|0}	OFF	

:SOURce:TUNE:DUT:VOLT

```
:SOURce:TUNE:DUT:VOLT <value>
:SOURce:TUNE:DUT:VOLT?
```

Sets/gets the voltage at the DUT TUNE port. Returns the configured value. If the output is turned off, it doesn't neccessarily return 0, as an internal voltage may be configured.

***RST** 0.000000000

:SOURce:TUNE:DUT:STAT

```
:SOURce:TUNE:DUT:STAT ON|OFF|1|0
:SOURce:TUNE:DUT:STAT?
```

Enables/disables the DUT TUNE port

***RST** 0

5.6 :STATus Subsystem

This subsystem controls the status-reporting structures.

Command	Parameters	Unit (default)	Remark
:STATus:OPERation[:EVENt]?			
:STATus:OPERation:CONDition?			
:STATus:OPERation:ENABle	<value>		
:STATus:OPERation:PTR	<value>		
:STATus:OPERation:NTR	<value>		
:STATus:PREset			
:STATus:QUEStionable[:EVENt]?			
:STATus:QUEStionable:CONDition?			
:STATus:QUEStionable:ENABle	<value>		
:STATus:QUEStionable:PTR	<value>		
:STATus:QUEStionable:NTR	<value>		

:STATus:OPERation[:EVENt]

`:STATus:OPERation[:EVENt]?`

Returns the contents of the operation status event register and clears it.

***RST** 0

:STATus:OPERation:CONDition

`:STATus:OPERation:CONDition?`

Returns the contents of the operation status condition register.

***RST** 0

:STATus:OPERation:ENABle

`:STATus:OPERation:ENABle <value>`

Sets the enable mask of the operation status event register.

:STATus:OPERation:PTR

`:STATus:OPERation:PTR <value>`

Sets the positive transition filter of the operation status event register.

:STATus:OPERation:NTR

`:STATus:OPERation:NTR <value>`

Sets the negative transition filter of the operation status event register.

:STATus:PRESet

`:STATus:PRESet`

Disables all status events, clears all negative transition filters and sets all positive transition filters.

:STATus:QUEStionable[:EVENt]

`:STATus:QUEStionable[:EVENt]?`

Returns the contents of the questionable status event register and clears it.

***RST** 0

:STATus:QUEStionable:CONDition

`:STATus:QUEStionable:CONDition?`

Returns the contents of the questionable status condition register.

***RST** 0

:STATus:QUEStionable:ENABle

`:STATus:QUEStionable:ENABle <value>`

Sets the enable mask of the questionable status event register.

:STATus:QUEStionable:PTR

`:STATus:QUEStionable:PTR <value>`

Sets the positive transition filter of the questionable status event register.

:STATus:QUEStionable:NTR

`:STATus:QUEStionable:NTR <value>`

Sets the negative transition filter of the questionable status event register.

5.7 :SYSTem Subsystem

Command	Parameters	Unit	Remark
:SYSTem:ERRor[:NEXT]?			
:SYSTem:ERRor:ALL?			
:SYSTem:PRESet			
:SYSTem:VERSion?			

:SYSTem:ERRor[:NEXT]

`:SYSTem:ERRor[:NEXT]?`

Return parameters: Integer error number. This query is a request for the next entry in the instrument's error queue. Error messages in the queue contain an integer in the range [-32768, 32768] denoting an error code and associated descriptive text.

***RST** 0,"No error"

:SYSTem:ERRor:ALL

`:SYSTem:ERRor:ALL?`

Return parameters: List of integer error numbers. This query is a request for all entries in the instrument's error queue. Error messages in the queue contain an integer in the range [-32768, 32768] denoting an error code and associated descriptive text. This query clears the instrument's error queue.

***RST** 0,"No error"

:SYSTem:PRESet

`:SYSTem:PRESet`

Resets most instruments functions to factory-defined conditions. This command is similar to the `*RST` command.

:SYSTem:VERSion

`:SYSTem:VERSion?`

Returns the SCPI version number that the instrument software complies with.

5.8 [:SYSTem:COMMunicate] Subsystem

Command	Parameters	Unit
:SYSTem:COMMunicate:LAN:CONFig	DHCP|MANual|AUTO	DHCP
:SYSTem:COMMunicate:LAN:DEFaults		
:SYSTem:COMMunicate:LAN:GATEway	<ipstring>	“0.0.0.0”
:SYSTem:COMMunicate:LAN:IP	<ipstring>	
:SYSTem:COMMunicate:LAN:RESTart		
:SYSTem:COMMunicate:LAN:SUBNet	<ipstring>	“255.255.255.0”

:SYSTem:COMMunicate:LAN:CONFig

```
:SYSTem:COMMunicate:LAN:CONFig DHCP|MANual|AUTO
:SYSTem:COMMunicate:LAN:CONFig?
```

Sets the instruments internet protocol (IP) address. The parameter can be one of the following:

`MANual`	The user assigns an IP address to the instrument.
`DHCP`	The network assigns an IP address to the instrument. If DHCP fails, manual configuration will be used.
`AUTO`	The network assigns an IP address to the instrument with a fallback to Auto-IP if DHCP fails. If both DHCP and Auto-IP fail, manual configuration will be used

***RST** AUTO

:SYSTem:COMMunicate:LAN:DEFaults

```
:SYSTem:COMMunicate:LAN:DEFaults
```

Restores the instrument's LAN settings to their factory default values.

:SYSTem:COMMunicate:LAN:GATEway

```
:SYSTem:COMMunicate:LAN:GATEway <ipstring>
:SYSTem:COMMunicate:LAN:GATEway?
```

Sets the gateway for local area network (LAN) access to the instrument from outside the current sub-network. The query returns the current setting, not the saved setting.

***RST** "192.168.1.1"

:SYSTem:COMMunicate:LAN:IP

```
:SYSTem:COMMunicate:LAN:IP <ipstring>
:SYSTem:COMMunicate:LAN:IP?
```

Sets the instrument's local area network (LAN) internet protocol (IP) address for your IP network connection.

:SYSTem:COMMunicate:LAN:RESTart

```
:SYSTem:COMMunicate:LAN:RESTart
```

Restarts the network to enable changes that have been made to the LAN setup.

:SYSTem:COMMunicate:LAN:SUBNet

```
:SYSTem:COMMunicate:LAN:SUBNet <ipstring>
:SYSTem:COMMunicate:LAN:SUBNet?
```

Sets the instrument's local area network (LAN) subnet mask address for your internet protocol (IP) network connection.

***RST** "255.255.255.0"

6 Examples

This chapter contains various commented command sequences that showcase the usage of the SCPI interface.

6.1 Absolute Phase Noise

6.1.1 Minimal Example

```
// measurement
> SENS:MODE PN                // select phase noise measurement
> INIT                        // start measurement
> CALC:WAIT:AVER ALL          // wait for the measurement to finish
> SYST:ERR:ALL?               // check if measurement was successful
< read error queue            // 0: ok, <0: fail (error code)
> CALC:PN:TRAC:SPOT? 1E6      // request spot noise value at 1MHz offset
< read value                  // value is in dBc/Hz
```

6.1.2 Full Configuration Example

```
// configuration
> SENS:MODE PN                // set phase noise measurement
> SENS:PN:REF NORM            // select standard internal references
> SENS:PN:LOB:AUTO ON         // enable automatic bandwidth selection
> SENS:PN:FREQ:AUTO ON        // enable frequency search
> SENS:PN:FREQ:DET ALW        //  ^ every time
> SENS:PN:KPHI:AUTO ON        // enable Kphi detection
> SENS:PN:KPHI:DET ALW        //  ^ every time
> SENS:PN:IFG:AUTO ON         // enable automatic gain selection
> SENS:PN:IFG:DET ALW         //  ^ every time
> SENS:PN:TEST O1e3,O1e6,F,J // define test set (PN@1kHz, PN@1MHz, freq, jitter)
> SENS:PN:RES                 // reset measurement configuration

// measurement
> SENS:PN:AVER 1              // 1 averages
> SENS:PN:CORR 10             // 10 correlations
> SENS:PN:PPD 150             // 150 points per decade
> SENS:PN:FREQ:STAR 10        // minimum offset frequency 10Hz
> SENS:PN:FREQ:STOP 50E6      // maximum offset frequency 50MHz
> SENS:PN:FUNC:RANG 12E3,5E6 // set integration interval to [12kHz - 5MHz]
> SENS:PN:SPUR:OMIS ON        // disable spurs (enable omission of spurs)
> SENS:PN:SMO:STAT 0          // disable smoothing
> INIT                        // trigger measurement start
loop
 > CALC:WAIT:AVER ALL,500     // wait for the measurement to finish
 < SYST:ERR:ALL?              // check if measurement was successful
                              // 0: ok, <0: fail (or timeout -393416 -> loop)
end loop

// retrieve full trace
> CALC:PN:TRAC:FREQ?          // request frequency data
< read list                   // binary format of list explained below
> CALC:PN:TRAC:NOIS?          // request phase noise data
< read list                   // binary format of list explained below

// retrieve test set
< CALC:TEST?                  // retrieve test set
```

6.2 VCO Characterization

6.2.1 Configuration Example

```
// configuration
> SENS:MODE VCO              // select VCO characterization
> SENS:VCO:TEST:FREQ ON      // enable frequency parameter
> SENS:VCO:TEST:ISUP ON      // enable supply current parameter
> SENS:VCO:TEST:KPUS ON      // enable pushing parameter
> SENS:VCO:TEST:KVCO ON      // enable Kvco parameter
> SENS:VCO:TEST:PN ON        // enable spot noise parameter
> SENS:VCO:TEST:PN:OFFS 1.2E3,1E5  // set two spot noise offsets: 1.2kHz, 100kHz
> SENS:VCO:TEST:POW ON       // enable power parameter

// measurement
> SENS:VCO:TYPE VCO          // set DUT Type (VCO or VCXO)
> SENS:VCO:VOLT:POIN 11      // set 11 measurement points
> SENS:VCO:VOLT:STAR 0.5     // set tuning range minimum to 0.5V
> SENS:VCO:VOLT:STOP 4.5      // set tuning range maximum to 10V
> SOUR:SUPP1:VOLT 5          // set supply voltage to 6V
> SOUR:SUPP1:STAT ON         // enable supply voltage
> INIT                       // trigger measurement start

loop
 > CALC:VCO:WAIT ALL,500     // wait for the measurement to finish
 > SYST:ERR:ALL?             // check if measurement was successful
 < read error queue          // 0: ok, <0: fail (or timeout -393416 -> loop)
end loop

// retrieve results
> CALC:VCO:TRAC:VOLT?        // request control voltage data array
< read list                  // binary format of list explained below
> CALC:VCO:TRAC:FREQ?        // request frequency data array
< read list                  // binary format of list explained below
> CALC:VCO:TRAC:KVCO?        // request Kvco data array
< read list                  // binary format of list explained below
> CALC:VCO:TRAC:KPUS?        // request pushing data array
< read list                  // binary format of list explained below
> CALC:VCO:TRAC:ISUP?        // request supply current data array
< read list                  // binary format of list explained below
> CALC:VCO:TRAC:POW?         // request power level data array
< read list                  // binary format of list explained below
> CALC:VCO:TRAC:PN? 1        // request spot noise data array @offset #1 (1.2kHz)
< read list                  // binary format of list explained below
```

6.3 Amplitude Noise

6.3.1 Minimal Example

```
// measurement
> SENS:MODE AN                // select amplitude noise measurement
> INIT                        // start measurement
> CALC:WAIT:AVER ALL          // wait for the measurement to finish
> SYST:ERR:ALL?               // check if measurement was successful
< read error queue            // 0: success, <0: failed
> CALC:AN:TRAC:SPOT? 1E6      // request spot noise value at 1MHz offset
< read value                  // value is in dBc/Hz
```

6.3.2 Configuration Example

```
// configuration
> SENS:MODE AN                // set amplitude noise measurement
> SENS:AN:FREQ:AUTO ON        // enable frequency search
> SENS:AN:FREQ:DET ALW        //  ^ every time
> SENS:AN:RES                 // reset measurement configuration

// measurement
> SENS:AN:AVER 1              // 1 averages
> SENS:AN:CORR 10             // 10 correlations
> SENS:AN:PPD 150             // 150 points per decade
> SENS:AN:FREQ:STAR 10        // minimum offset frequency 10Hz
> SENS:AN:FREQ:STOP 40E6      // maximum offset frequency 40MHz
> SENS:AN:SPUR:THR 15         // spur threshold 15dB
> SENS:AN:SPUR:OMIS ON        // enable omission of spurs (don't show spurs)
> SENS:AN:SMO:APER 5          // smoothing aperture 5%
> SENS:AN:SMO:STAT ON         // enable smoothing
> INIT                        // trigger measurement start

loop
 > CALC:WAIT:AVER ALL,500     // wait for the measurement to finish
 > SYST:ERR:ALL?              // check if measurement was successful
 < read error queue           // 0: ok, <0: fail (or timeout -393416 -> loop)
end loop

// retrieve full trace
> CALC:AN:TRAC:FREQ?          // request frequency data
< read list                   // binary format of list explained below
> CALC:AN:TRAC:NOIS?          // request amplitude noise data
< read list                   // binary format of list explained below

// retrieve spot noise values
< CALC:AN:TRAC:SPOT? 1E3      // get spot noise value at 1kHz offset
```

6.4 Absolute Phase Noise (Mode FN)

6.4.1 Minimal Example

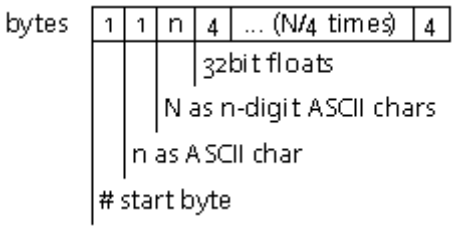
```
// measurement
> SENS:MODE FN               // select FN mode phase noise measurement
> INIT                       // start measurement
> CALC:WAIT:AVER ALL         // wait for the measurement to finish
> SYST:ERR:ALL?              // check if measurement was successful
< read error queue           // 0: success, <0: failed
> CALC:FN:TRAC:SPOT? 1E6     // request spot noise value at 1MHz offset
< read value                 // value is in dBc/Hz
```

6.4.2 Configuration Example

```
// configuration
> SENS:MODE FN                // set FN mode phase noise measurement
> SENS:FN:FREQ:AUTO ON        // enable frequency search
> SENS:FN:FREQ:DET ALW        //  ^ every time
> SENS:FN:RES                 // reset measurement configuration

// measurement
> SENS:FN:AVER 1              // 1 averages
> SENS:FN:CORR 10             // 10 correlations
> SENS:FN:PPD 150             // 150 points per decade
> SENS:FN:FREQ:STAR 10        // minimum offset frequency 10Hz
> SENS:FN:FREQ:STOP 40E6      // maximum offset frequency 40MHz
> SENS:FN:SPUR:THR 15         // spur threshold 15dB
> SENS:FN:SPUR:OMIS ON        // enable omission of spurs (don't show spurs)
> SENS:FN:SMO:APER 5          // smoothing aperture 5%
> SENS:FN:SMO:STAT ON         // enable smoothing
> INIT                        // trigger measurement start

loop
 > CALC:WAIT:AVER ALL,500     // wait for the measurement to finish
 > SYST:ERR:ALL?              // check if measurement was successful
 < read error queue           // 0: ok, <0: fail (or timeout -393416 -> loop)
end loop

// retrieve full trace
> CALC:FN:TRAC:FREQ?          // request frequency data
< read list                   // binary format of list explained below
> CALC:FN:TRAC:NOIS?          // request phase noise data
< read list                   // binary format of list explained below

// retrieve spot noise values
< CALC:AN:TRAC:SPOT? 1E3      // get spot noise value at 1kHz offset
```

7 Block Data Format

The block data format is used to transfer an array of floating point values via the SCPI protocol. It follows the definition of block data according to IEEE 488.2. It contains a header and a block of 32bit floats. The header contains a start byte, a byte containing the number *n* defining the number of 1-byte digits following in the header. The *n* following 1-byte digits define the number of 4-byte floats following in the body of the package. See the color coded example below.



Example: 0x23323132 0050C347 79689A48 00247449

0x23: ASCII code for # -> start

0x32: ASCII code for 2 -> n=2

0x3132: ASCII code for 1 (0x31) and 2 (0x32) -> N=12 (3x 32bit float values)

0x0050C347: 32bit float -> 100000.0

0x79689A48: 32bit float -> 316227.78125

0x00247449: 32bit float -> 1000000.0