Programmer’s Manual V1.2 (prelim)

APPH Phase Noise Measurement Systems

Models APPH6040, APPH20G

WARRANTY

All Anapico instruments are warranted against defects in material and workmanship for a period of two years from the date of shipment. Anapico will, at its option, repair or replace products that prove to be defective during the warranty period, provided they are returned to Anapico 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. Anapico 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 APPH Signal Source Analyzers / Phase Noise Measurement Systems using commands sent from an external controller via remote control. 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 APPH 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 APPH

The APPH can be accessed though 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 APPH 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 APPH 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.

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 APPH 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 APPH 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 APPH 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 support@anapico.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 signal generator. 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 signal generator. A connection, established between computer and signal generator, 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 APPH and other USB compatible devices must be connected via the USB interface port (type B)

The APPH 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 APPH devices. NI-VISA is configured to detect USBTMC compliant instruments such as the APPH. 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 Anapico API

Anapico API programming interface supports direct communication to APPH using Anapico's proprietary DLL driver libraries. The library allows setup a communication channel though USB, LAN, or GPIB from any programming environment.
Please contact Anapico 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 APPH from an external computer. The computer sends commands and instructions to the APPH and receives data sent from the APPH via GPIB.

You can connect up to 15 devices in a single GPIB system.
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 APPH

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 the chapter 4 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 instrument options currently installed on the signal generator.

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 APPH 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 APPH 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 APPH. 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 can be a mixture of upper and lower case characters. 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 upper case and the remainder in lower case. Unrecognized versions of long form or short form commands, or improper syntax, will generate an error.

Structure of a Command Line

A command line may consist of one or several commands. It is terminated by 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. 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.

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. ON or OFF is returned by a query.

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 APPH 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 phasenoise 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 intodata 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 is appears in the SCPI tree as TRIGger, ARM, INITiate, and ABORt subsystems.

:UNIT

All APPH 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 APPH implements the status byte register, the Service Request Enable Register, the Standard Event Status Register, and the Standard Event Status Enable Register.

The APPH 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 APPH. 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 APPH 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 APPH 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 APPH event or condition. The bits in the Summary Status Byte provide the following:

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.

4.8 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 APPH 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)

4.9 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. The bits in the Operation Event register provide the following:

4.10 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 (default)	Remark
:ABORt			

:ABORt

This command causes measurement or sweep in progress to abort. 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.

The subsystem works for BB (baseband noise), and PN (phase noise), respectively.

Command	Parameters	Unit (default)	Remark
:CALCulate:FREQuency?		Hz	
:CALCulate:POWer?		dBm	
:CALCulate:XX:TRACe:FREQuency?		Hz	binary list
:CALCulate:PN:TRACe:NOISe?		dBc/Hz	binary list
:CALCulate:BB:TRACe:NOISe?		dBV/Hz	FW: 0.5.1
:CALCulate:PN:TRACe:SPOT?		dBc/Hz	
:CALCulate:BB:TRACe:SPOT?		dBV/Hz	FW: 0.5.1
:CALCulate:XX:TRACe:SPURious:FREQuency?		Hz	binary list
:CALCulate:XX:TRACe:SPURious:POWer?		dBc	binary list
:CALCulate:XX:TRACe:FUNCtion:JITTer?		s	
:CALCulate:XX:TRACe:FUNCtion:INTegral?		dBc	
:CALCulate:XX:TRACe:FUNCtion:AVARiance:TAU?			FW: 0.5.1

:CALCulate:XX:TRACe:FUNCtion:AVARiance:SIGMa?			FW: 0.5.1
:CALCulate:WAIT:AVERage	{NEXT | ALL | <value>}[,<value>]		
:CALCulate:XX:PREL:AVER?			

:CALCulate:FREQuency?

This command reads out the internal frequency counter and returns the detected DUT frequency.

:CALCulate:POWer?

This command reads out the internal power meter and returns the detected DUT power.

:CALCulate:XX:TRACe:FREQuency?

This command returns a binary list of offset frequency points in Hz of the active measurement. The format of the list can be found in the chapter Examples.

:CALCulate:XX:TRACe:NOISe?

This command returns the list of spectrum points. The unit is:

- Phasenoise measurement: dBc/Hz
- FFT analyzer measurement: dBV/Hz

The format is equal to the CALC:XX:TRAC:FREQ? query.

:CALCulate:XX:TRACe:SPOT?

This command returns the spot noise at the specified spot noise offset frequency. The unit is the same as the CALC:XX:TRAC:NOIS? query.

:CALCulate:XX:TRACe:SPURious:FREQuency?

This command returns an array of offset frequency points in Hz for the spurious list of the active trace.

:CALCulate:XX:TRACe:SPURious:POWer?

This command returns an array of power values in dBc/Hz (or dBV/Hz for an FFT analyzer measurement) for the spurious list of the active trace.

:CALCulate:PN:TRACe:FUNCtion:JITTer?

Gets the total RMS jitter of the active trace.

:CALCulate:PN:TRACe:FUNCtion:INTegral?

Gets the integral noise of the active trace.

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

This command gets a binary list of tau values of the Allan Variance, for the active trace.

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

This command gets a binary list of sigma values of the Allan Variance, for the active trace.

: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). 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?.

:CALCulate:XX:PREL:AVERage?

Gets the number of iterations for the current data saved on the device.

5.3 :INITiate Subsystem

The :INITiate subsystem controls the state of the APPH 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 (default)	Remark
:INITiate[:IMMediate]			
:INITiate:CONTinuous	{ON|OFF|1|0}	OFF	FW: 0.5.1

:INITiate[:IMMediate]

Sets APPH trigger to the armed state.

:INITiate:CONTinuous ON|OFF|1|0

Continuously rearms the APPH trigger system after completion of a triggered sweep.

5.4 :SENSe Subsystem

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

Command	Parameters	Default	Remark
:SENSe:FREQuency	<value>	100e6 Hz	
:SENSe:FREQuency:EXECute			
:SENSe:FREQuency:AUTO	{ON | OFF | 1 | 0}	ON	
:SENSe:FREQuency:DETect	{ALWays | ONCe | NEVer}	ALW	
:SENSe:POWer	<value>	0 dBm	
:SENSe:POWer:EXECute			
:SENSe:POWer:AUTO	{ON | OFF | 1 | 0}	ON	
:SENSe:POWer:DETect	{ALWays | ONCe | NEVer}	ALW	

:SENSe:MODE	{PN | TRAN | BB}	PN	
: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:REFerences	{INTernal | EXTernal}	INT	
:SENSe:PN:REFerences:SENSitivity	{1|2},<value>	1 Hz / V	
:SENSe:PN:REFerences:SENSitivity:EXEcute			FW: 0.5.1
:SENSe:PN:REFerences:TUNE:MAX	{1|2},{3 ~ 20}	3 V	FW: 0.5.1
:SENSe:RESet			
:SENSe:XX:METHod	{SINGle | CC}	CC	
:SENSe:XX:AVERage	{1 ~ 10k}	1	
:SENSe:XX:ASET|ATTenuation	{0 ~ 30}	0 dB	
:SENSe:XX:ASET|ATTenuation:AUTO	{ON | OFF | 1 | 0}	ON	
:SENSe:XX:ASET|ATTenuation:DETect	{ALWays | ONCe | NEVer}	ALWays	
:SENSe:XX:IFGain	<value>	46 dB	
:SENSe:XX:IFGain:AUTO	{ON | OFF | 1 | 0}	ON	
:SENSe:XX:IFGain:DETect	{ALWays | ONCe | NEVer}	ALWays	
:SENSe:XX:FREQuency:STARt	{ 0.1 | 0.5 | 1 | 10 | 100 | 1k | 10k | 100k }	10 Hz	
:SENSe:XX:FREQuency:STOP	{1k | 10k | 100k | 1M |10M | 50M }	50e6 Hz	
:SENSe:XX:FREQuency:SPOT	{1 ~ 10M}	100 Hz	
:SENSe:XX:FUNCtion:RANGe	{0.1 ~ 50M},{0.1 ~ 50M}	10,50E6 Hz	
:SENSe:XX:PPD	{1 ~ 500}	250	
:SENSe:XX:SPURious:OMISsion	{ON | OFF | 1 | 0}	ON	FW: 0.5.1
:SENSe:XX:SPURious:THReshold	{1 ~ 70}	10 dB	FW: 0.5.1
:SENSe:XX:SMOothing:APERture	{0.05 ~ 20}	0.05 %	FW: 0.5.1
:SENSe:XX:SMOothing:STATe	{ON | OFF | 1 | 0}	OFF	FW: 0.5.1

:SENSe:FREQuency <value>

:SENSe:FREQuency?

Sets/Gets the DUT frequency in Hz.

:SENSe:FREQuency:EXECute

Start the frequency search. See the :CALC subsystem on how to read out the measurement.

:SENSe:FREQuency:AUTO ON|OFF

:SENSe:FREQuency:AUTO?

Enable/Disable the automatic frequency search at the start of the measurement.

:SENSe:FREQuency:DETect ALWays|ONCe|NEVer

:SENSe:FREQuency:DETect?

Sets/Gets how often the automatic frequency search should be performed. 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.

:SENSe:POWer <value>

:SENSe:POWer?

Sets the DUT power in dBm.

:SENSe:POWer:EXECute

Start the power measurement. See the :CALC subsystem on how to read out the measurement.

:SENSe:POWer:AUTO ON|OFF

:SENSe:POWer:AUTO?

Enable/Disable the automatic power measurement at the start of the measurement.

:SENSe:POWer:DETect ALWays|ONCe|NEVer

:SENSe:POWer:DETect?

Sets how often the automatic power measurement should be performed. 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.

:SENSe:MODE PN|TRAN|BB

Sets the active measurement mode. PN: phasenoise; TRAN: transient measurement (not yet available); BB: FFT analyzer mode (not yet available).

:SENS:PN:KPHI <value>

:SENS:PN:KPHI?

This command sets/gets Kphi in rad / V.

:SENS:PN:KPHI:AUTO ON|OFF

:SENS:PN:KPHI:AUTO?

Enable/Disable the Kphi detection at the start of the measurement.

:SENS:PN:KPHI:DETect ALWays|ONCe|NEVer

:SENS: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 performed yet since startup of the instrument; NEVer: Always skip it.

:SENSe:PN:LOBandwidth {0.1 ~ 10k}

:SENSe:PN:LOBandwidth?

This command gets/sets the PLL bandwidth for the selected channel.

:SENSe:PN:LOBandwidth:AUTO ON|OFF

:SENSe:PN:LOBandwidth:AUTO?

Enable/Disable the automatic selection of the optimal bandwidth during the measurement.

:SENSe:PN:REFerences INT|EXT

:SENSe:PN: REFerences?

This command selects/gets the reference used for the measurement.

:SENSe:PN:REFerences:SENSitivity <value>

:SENSe:PN: REFerences:SENSitivity?

This command sets/gets the reference sensitivity.

:SENSe:PN:REFerences:SENSitivity:EXECute

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

:SENSe:PN:REFerences:TUNE:MAX {1|2},{3 ~ 20}

Sets the maximum tuning voltage of the selected reference (first parameter) to the specified value in volts. The minimum tuning voltage is fixed at 0 V.

:SENS:RESet

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

:SENSe:XX:METHod SINGle | CC

Sets the active measurement configuration. This command affects measurements with external references sources only.

:SENSe:XX:AVERage {1 ~ 10k}

:SENSe:XX:AVERage?

This command sets/gets average count of the measurement.

:SENSe:XX:ASET {0 ~ 30}

:SENSe:XX:ASET?

This command sets/gets the RF step attenuator for the selected channel in dB.

:SENSe:XX:ASET:AUTO ON|OFF

:SENSe:XX:ASET:AUTO?

Enables/Disables the automatic step attenuator.

:SENSe:XX:ASET:DETect ALWays|ONCe|NEVer

:SENSe:XX:ASET:DETect?

Sets how often the automatic step attenuator should be 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.

:SENSe:XX:IFGain <value>

:SENSe:XX:IFGain?

This command sets/gets the IF gain in dB.

:SENSe:XX:IFGain:AUTO ON|OFF

:SENSe:XX:IFGain:AUTO?

Enables/Disables the automatic IF gain setting.

:SENSe:XX:IFGain:DETect ALWays|ONCe|NEVer

:SENSe:XX: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.

:SENSe:XX:FREQuency:STARt <value>

:SENSe:XX:FREQuency:STARt?

This command sets/gets start offset frequency.

:SENSe:XX:FREQuency:STOP <value>

:SENSe:XX:FREQuency:STOP?

This command sets/gets stop offset frequency.

:SENSe:PN:FREQuency:SPOT <value>

:SENSe:PN:FREQuency:SPOT?

This command sets/gets the desired offset frequency spot to be measured.

:SENSe:XX:FUNCtion:RANGe <value>,<value>

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

:SENSe:XX:PPD <value>

:SENSe:XX:PPD?

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

:SENSe:TRACe:SPURious:OMISsion ON|OFF

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

:SENSe:TRACe:SPURious:THREshold <value>

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:TRACe:SMOothing:APERture <value>

Sets the smoothing aperture of the trace in %.

:SENSe:TRACe:SMOothing:STATe ON|OFF

Enables/Disables trace smoothing.

5.5 :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>		

:OPERation?

:STATus:OPERation[:EVENt]?

This query returns the contents of the operation status event register and clears it.

:OPERation:CONDition?

:STATus:OPERation:CONDition?

This query returns the contents of the operation status condition register.

:OPERation:ENABle

:STATus:OPERation:ENABle

This command sets the enable mask of the operation status event register.

:OPERation:PTR

:STATus:OPERation:PTR

This command sets the positive transition filter of the operation status event register.

:OPERation:NTR

:STATus:OPERation:NTR

This command sets the negative transition filter of the operation status event register.

:PRESet

:STATus:PRESet

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

:QUEStionable?

:STATus:QUEStionable [:EVENt]?

This query returns the contents of the questionable status event register and clears it.

:QUEStionable:CONDition?

:STATus:QUEStionable:CONDition?

This query returns the contents of the questionable status condition register.

:QUEStionable:ENABle

:STATus:QUEStionable:ENABle

This command sets the enable mask of the questionable status event register.

:QUEStionable:PTR

:STATus:QUEStionable:PTR

This command sets the positive transition filter of the questionable status event register.

:QUEStionable:NTR

:STATus:QUEStionable:NTR

This command sets the negative transition filter of the questionable status event register.

5.6 :SYSTem Subsystem

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

:ERRor?

:SYSTem:ERRor[:NEXT]?

Return Parameters: Integer error number

Query command 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.

:ERRor:ALL?

:SYSTem:ERRor:ALL?

Return Parameters: List of integer error number

Query command is a request for the 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.

:PRESet

:SYSTem:PRESet

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

:VERSion?

:SYSTem:VERSion?

Returns the SCPI version number that the instrument software complies with [1999.0]

5.7 [:SYSTem:COMMunicate] Subsystem

Command	Parameters	Unit (default)
: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"

:LAN:CONFig

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

:SYSTem:COMMunicate:LAN:CONFig?

This command sets the signal generator's internet protocol (IP) address.

MANual The user assigns an IP address to the signal generator.

DHCP The network assigns an IP address to the signal generator. If DHCP fails, manual configuration will be used.

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

:LAN:DEFaults

:SYSTem:COMMunicate:LAN:DEFaults

This command restores the instrument's LAN settings to their factory default values.

:LAN:GATeway

:SYSTem:COMMunicate:LAN:GATeway <ipstring>

:SYSTem:COMMunicate:LAN:GATeway?

This command sets the gateway for local area network (LAN) access to the signal generator from outside the current sub- network. The query returns the current setting, not the saved setting.

:LAN:IP

:SYSTem:COMMunicate:LAN:IP <ipstring>

:SYSTem:COMMunicate:LAN:IP?

This command sets the signal generator's local area network (LAN) internet protocol (IP) address for your IP network connection.

:LAN:RESTart

:SYSTem:COMMunicate:LAN:RESTart

This command restarts the network to enable changes that have been made to the LAN setup.

:LAN:SUBNet

:SYSTem:COMMunicate:LAN:SUBNet <ipstring>

:SYSTem:COMMunicate:LAN:SUBNet?

This command sets the signal generator's local area network (LAN) subnet mask address for your internet protocol (IP) network connection.

5.8 :TRIGger Subsystem

The trigger system is responsible for tasks such as detecting the start of a measurement cycle (triggering) and enabling/disabling measurement for each measurement.

A trigger signal comprises both positive and negative signal transitions (states), which are also called high and low periods. You can configure the APPH to trigger on either state of the trigger signal. It is common to have multiple triggers, also referred to as trigger occurrences or events, occur when the instrument requires only a single trigger. In this situation, the APPH recognizes the first trigger and ignores the rest.

There are four parts to configuring the trigger:

1. Choosing the trigger type which controls the waveform's transmission.
    - NORMal : trigger edge initiates/stops sweeps

- GATE : trigger level starts/stops sweep

2. Setting the waveform's response to triggers:
   - CONTinuous : reapeatedly accepts trigger events
   - SINGle : uses only one trigger event

3. Selecting the trigger source which determines how the APPH receives its trigger signal, internally or externally. The GATE choice requires an external trigger.

4. Setting the trigger polarity when using an external source

Command	**Parameters**	**Unit (default)**
[SOURce]:TRIGger[:SEQuence]:TYPE	NORMal | GATE | POINT	
[SOURce]:TRIGger[:SEQuence]:TYPE:GATE	LOW|HIGH	HIGH
[SOURce]:TRIGger[:SEQuence]:SOURce	IMMediate | EXT | BUS	IMM
[SOURce]:TRIGger[SEQuence]:SLOPe	POSitive|NEGative	POS

:TRIGger:TYPE

[SOURce]:TRIGger:TYPE NORMal | GATE | POINT

[SOURce]:TRIGger:TYPE?

This command sets the trigger type

The following list describes the trigger type command choices:

NORMal	Upon triggering, the waveform sequence plays according to settlings defined by :INITiate:CONTinuous (only once or repeatedly)
GATE	An external trigger signal repeatedly starts and stops the waveform's playback. The time duration for playback depends on the duty period of the trigger signal and the gate polarity selection. The waveform plays during the inactive state and stops during the active polarity selection state. The active state can be set high or low. The gate mode works only with an external trigger source.
POINT	Upon triggering, only a single point of the sweep (list) is played.

***RST** NORM

:TRIGger:TYPE:GATE

[SOURce]:TRIGger:TYPE:GATE LOW|HIGH

[SOURce]:TRIGger:TYPE:GATE?

This command selects the active state (gate polarity) of the gate while using the gating trigger mode. The LOW and HIGH selections correspond to the low and high states of an external trigger signal. For example, when you select HIGH, the active state occurs during the high of the trigger signal.
When the active state occurs, the APPH starts the waveform playback at the last played sample point, then stops the playback at the next sample point when the inactive state occurs.

LOW The waveform playback starts when the trigger signal goes low (active state) and stops when the trigger signal goes high (inactive state).

HIGH The waveform playback starts when the trigger signal goes high (active state) and stops when the trigger signal goes low (inactive state).

***RST** HIGH

:TRIGger[SEQuence]SOURce

[SOURce]:TRIGger[SEQuence]:SOURce IMMediate | EXTernal | BUS

[SOURce]:TRIGger[SEQuence]:SOURce?

This command sets the trigger source.

IMMediate No waiting for a trigger event occurrs

EXTernal This choice enables the triggering of a sweep event by an externally applied signal at the MOD IN connector.

BUS This choice enables triggering over the LAN using the *TRG commands.

***RST** IMM

:TRIGger[SEQuence]:SLOPe

[SOURce]:TRIGger[SEQuence]:SLOPe POSitive|NEGative

[SOURce]:TRIGger[SEQuence]:EXTernal:SLOPe?

This command sets the polarity for an external trigger signal while using the continuous, single triggering mode. The POSitive and NEGative selections correspond to the high (positive) and low (negative) states of the external trigger signal. For example, when you select POSitive, the waveform responds (plays) during the high state of the trigger signal. When the APPH receives multiple trigger occurrences when only one is required, the instrument uses the first trigger and ignores the rest.

***RST** POS

5.9 UNIT Subsystem

Command	Parameters	Unit (default)	Remark
UNIT:POWer	W|V|DBM|DBC/HZ|UV/SQHZ	DBM	
UNIT:FREQuency	HZ|MHZ|GHZ	HZ	
UNIT:NOISe	NVSQHZ | DBMHZ	NVSQHZ	FW 1.0

UNIT:POWer

UNIT:POWer W|V|DBM|DBC/HZ|UV/SQHZ

*RST DBC/HZ

UNIT:FREQuency

UNIT:FREQuency HZ|MHZ|GHZ

*RST HZ

UNIT: NOISe

UNIT: NOISe NVSQHZ | DBMHZ

*RST NVSQHZ

6 Examples (NEEDS REWORK)

6.1 Remote Phase Noise Measurement (FW 0.5.0)

Following the SCPI command sequence for a very simple remote phase noise measurement.

```
// measurement
> SENS:PN:AVER 10          // 10 averages
> SENS:PN:FREQ:STAR 10     // minimum offset frequency 10Hz
> SENS:PN:FREQ:STOP 50E6   // maximum offset frequency 50MHz
> 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:PN:TRAC:FREQ?       // request frequency data
< read list                // binary format of list explained below
> CALC:PN:TRAC:NOIS?       // request phasenoise data
< read list                // binary format of list explained below
```

Following the SCPI commands of a configured measurement with preliminary results every second.

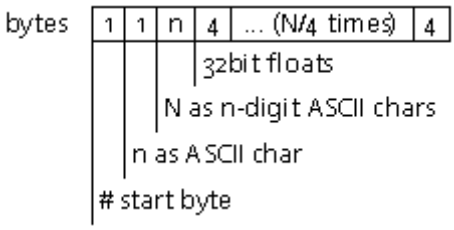
```
// configuration
> SENS:MODE PN             // set phase noise measurement
> SENS:PN:REF INT          // select internal references
> SENS:PN:LOB:AUTO ON      // enable automatic bandwidth selection
> SENS:FREQ:AUTO ON        // enable frequency search
> SENS:FREQ:DET ONC        //  ^ only once, then take previously determined value
> SENS:PN:IFG:AUTO ON      // enable frequency search
> SENS:PN:IFG:DET ALW      //  ^ every time
> SENS:PN:ASET:AUTO ON     // enable automatic input attenuation
> SENS:PN:ASET:DET ALW     //  ^ every time
> SENS:PN:KPHI:AUTO ON     // enable Kphi detection
> SENS:PN:KPHI:DET ALW     //  ^ every time
> SENS:RES                 // reset measurement configuration (f.e. the once-flags)

// measurement
> SENS:PN:AVER 10          // 10 averages
> SENS:PN:FREQ:STAR 10     // minimum offset frequency 10Hz
> SENS:PN:FREQ:STOP 50E6   // maximum offset frequency 50MHz
> INIT                     // start measurement

> CALC:WAIT:AVER NEXT      // request the next iteration as preliminary data
> SYST:ERR:ALL?            // check if measurement was successful
< read error queue         // 0: success, <0: failed
> CALC:PN:TRAC:FREQ?       // request preliminary frequency data
< read list                // binary format of list explained below
> CALC:PN:TRAC:NOIS?       // request preliminary phasenoise data
< read list                // binary format of list explained below

> CALC:WAIT:AVER ALL       // wait for the measurement to finish
> CALC:PN:TRAC:FREQ?       // request frequency data
< read list                // binary format of list explained below
> CALC:PN:TRAC:NOIS?       // request phasenoise data
< read list                // binary format of list explained below
```

Binary format of list



Example: 0x2332313200 50C34779689A4800247449

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