GE

Sensing & Inspection Technologies

PACE

Pressure Automated Calibration Equipment

SCPI Remote Communications Manual - K0472

Introduction

This technical manual provides SCPI protocol instructions for the remote control of the PACE Series indicators and controllers.

Safety

- The manufacturer has designed this product to be safe when operated using the procedures detailed in this manual. Do not use this product for any other purpose than that stated.
- This publication contains operating and safety instructions that must be followed to make sure of safe operation and to maintain the equipment in a safe condition. The safety instructions are either warnings or cautions issued to protect the user and the equipment from injury or damage.
- Use qualified* programming technicians and good engineering practice for all procedures in this publication.

❐ **Pressure**
Do not apply pressure greater the maximum safe working pressure to the PACE Series.

❐ **Maintenance**
The PACE Series must be maintained using the manufacturer's procedures and should be carried out by authorised service agents or the manufacturer's service departments.

❐ **Technical Advice**
For technical advice contact the manufacturer or subsidiary.

* A programming technician must have the necessary specialist knowledge of programming, technical knowledge and documentation to carry out the required work on the PACE Series.

Associated Documents:

A beginners Guide To SCPI by Barry Eppler, Published by Addison-Wesley Publishing Company Inc. for Hewlett Packard (ISBN 0-201-56350-9)

Table of Contents

The following abbreviations are used in this manual; abbreviations are the same in the singular and plural.

a	Absolute
ASCII	American Standard Code for Information Interchange
e.g.	For example
Fig.	Figure
ft	Foot
g	Gauge
GPIB	General purpose interface bus
i.e.	That is
IEEE 488	Institute of Electrical and Electronic Engineers standard 488 (for programmable devices with a digital interface)
m	Metre
max	Maximum
mbar	Millibar
min	Minute or minimum
No.	Number
RS232	Serial communications standard
Rx	Receive data
SCPI	Standard commands for programmable instruments
Tx	Transmit data
+ve	Positive
-ve	Negative
°C	Degrees Celsius
°F	Degrees Fahrenheit

The following units are used in this manual

ATM	atmosphere
BAR	bar
CMH2O	centimetres of water at 20°C
CMHG	centimetres of mercury
FTH2O	feet of water at 20°C
FTH2O4	feet of water at 4°C
HPA	hecto Pascals
INH2O	inches of water at 20°C
INH2O4	inches of water at 4°C
INH2O60	inches of water at 60°F
INHG	inches of mercury
KG/CM2	kilogrammes per square centimetre
KG/M2	kilogrammes per square metre
KPA	kilo Pascals
LB/FT2	pounds per square foot
MH2O	metres of water
MHG	metres of mercury
MMH2O	millimetres of water
MMHG	millimetres of mercury
MPA	mega Pascals
PA	Pascals
PSI	pounds per square inch
TORR	torr
MBAR	millibar

1 INTRODUCTION

1.1 General

The IEEE 488 and RS232 interfaces of the PACE Series provide remote control of the instrument from a suitable computer or controller. The SCPI protocol enables any instrument with a SCPI facility to be controlled using the same commands. The PACE Series use the full SCPI command set and the defined SCPI syntax.

The following sections describe and define each instrument command used by the PACE Series. The commands for the aeronautical option and the sensor calibration module option are described and defined in separate sections. Each section contains a quick reference structure of the relevant commands.

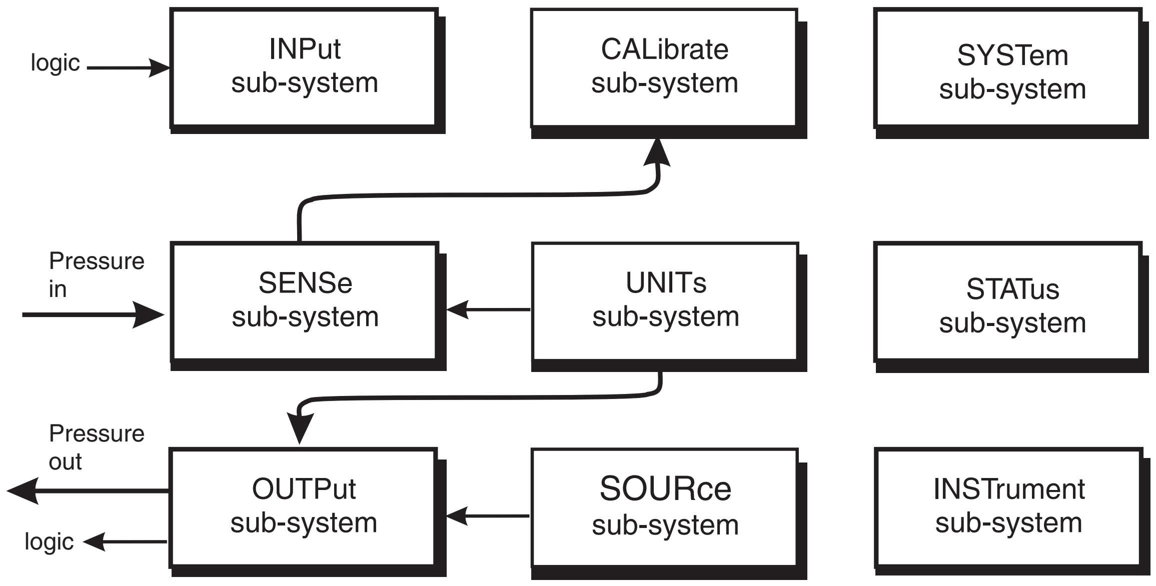


Figure 1-1 System Model

System Model

SCPI starts with a high-level block diagram of the measurement functions of the instrument. Each functional block is broken down into smaller block diagrams. SCPI contains a hierarchy of commands called a subsystem that maps directly to the hierarchy of the block diagram.

1.2 Remote/Local Operation

Most commands received over the SCPI interface automatically puts the PACE Series into remote control mode and disables the front panel touch-screen. Sending the LOC command returns the PACE Series to local control mode and enables the front panel touch-screen .

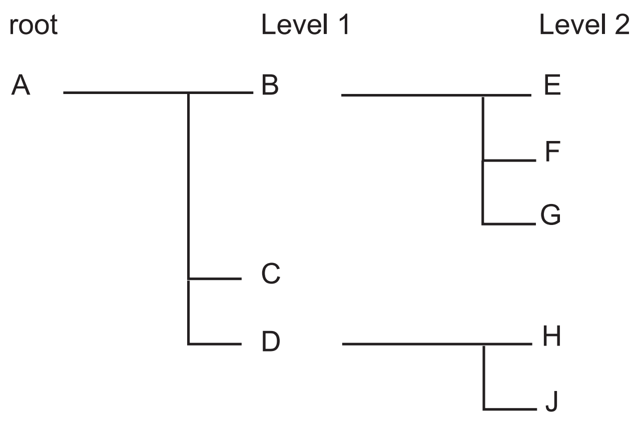
intentionally left blank

2 COMMAND STRUCTURE

This section describes the structure of the commands and data sent and received by a PACE Series Controller/Calibrator.

2.1 Notation

All SCPI commands are based on a hierarchical tree structure consisting of key words and parameters. Associated commands are grouped together under a common node in the hierachy.



In the command tree the command A is the root command. A tree pointer is used to decode the SCPI commands. At power-up the pointer goes to the root command.

2.2 Message Terminators

All SCPI commands are terminated by line feed i.e., either <newline> (ASCII character, decimal 10), EOI for IEEE. After receiving a termination character the tree pointer returns to the root command.

Colon

A colon moves the current path down one level in the command tree, (e.g., the colon in SOURCE:PRESSURE specifies PRESSURE the is one level below SOURCE). When the colon is the first character of the command, it specifies that the next command is a root level command (e.g., :SOURCE specifies that SOURCE is a root level command).

Semicolon

A semicolon separates two commands in the same message without changing the tree pointer.

(e.g., with reference to the tree):A:B:E;F;G

This equivalent to sending three messages:

```
:A:B:E
:A:B:F
:A:B:G
```

Commas

If a command requires more than one parameter, separate adjacent parameters by using a comma. A commas does not affect the tree pointer.

(e.g.) :SYSTEM:TIME 10,25,30

To execute a command the full path to the command must be specified:

(e.g.) :OUTPut:STATe ON

This turns the pressure controller on.

Note:

There must be a space between the command words and the parameter. In the above example there is a space between :STATe and ON.

SCPI commands are not case sensitive and may have a short form. In this manual, upper case letter identify the short form.

(e.g.) :OUTP is the short form of OUTPUT.

Some nodes can be the default node and these key words are optional when programming the command. The instrument processes the command, with the same effect, with or without the option node. In this manual [] enclose [default notes].

(e.g.) :SOURce[:PRESsure:][:LEVel][:IMMediate][:AMPlitude] 5.0

can be sent as

```
:SOURce:PRESsure:LEVel 5.0
```

or

```
:SOURce 5.0
```

This sets the set-point to 5.0

2.3 Program Headers

Program headers are keywords that identify a command, instruments accept both upper and lower case characters in a program header. There are two types of program header, common command headers and instrument control headers; each header can be a command or a query.

Common Command and Query Headers

The common command and query program header syntax, specified in IEEE 488.2, are defined as follows:

Command

```
*<PROGRAM MNEMONIC>
```

Query

```
*<PROGRAM MNEMONIC>?
```

Instrument Control Command and Query Headers

The instrument control command and query program header syntax controls and extracts data from the instrument as follows:

Command

```
:<MNEMONIC>
:<MNEMONIC> <PARAMETER>
```

Query

```
:<MNEMONIC>?
```

Instrument command headers can have a numeric suffix to identify each of several cases of the same header; the numeric suffix applies to both the long and short forms. All commands headers without a numeric suffix assume the value 1.

e.g.,

```
:OUTPut:LOGic1?
```

is the same as

```
:OUTPut:LOGic:?
```

Queries

A query is a program header with an attached question mark character (?). On receiving a query, the current settings are loaded in the output buffer. A query does not affect the operation or set-up of the instrument.

When the parameter of a command contains enumerated character data, both long form and short form are recognised. A query causes the return of data in the short form.

Querying numeric parameters causes the resulting data to be returned in the units selected by the instrument unless specified otherwise.

2.4 SCPI Data Types

A variety of data types can be sent to the instrument as parameters or sent out from the instrument as response data.

Decimal Data

All normal decimal expressions are accepted including optional signs, decimal point and scientific notation.

Note:

This includes floating point data.

The following are valid:

```
123
45.67
-2.6
4.6e-10
.76
```

A suffix multiplier can be added to the numeric value.

```
:SOUR 100 m
```

would set the programmable output to 0.1 units (100m units).

The multipliers supported are:

Mnemonic	Multiplier
A	1e-18
G	1e+9
K	1e+3
M	1e-3
T	1e+12

If a real value is sent to the instrument when an integer is expected, it will be rounded to an integer.

Integer Data

Integer data are whole numbers (containing no decimal places). A query of an integer value returns numbers containing no decimal places.

Note:

Integer values can be specified in binary, octal or hexadecimal formats using the suffix letters (upper or lower case) B, Q and H respectively.

e.g.,	#B1010	binary representation of 10
	#Q71	octal representation of 57
	#HFA	hexadecimal representation of 250

Hexadecimal digits A-F can be in upper or lower case.

Enumerated Character Data
Enumerated characters are used for data that has a finite number of values; enumerated parameters use mnemonics to represent each valid setting.
The mnemonics have long and short forms just like command mnemonics.

Example:

```
                :SOURce:PRESsure:SLEW:MODE MAXimum
```

selects the maximum rate mode.

A query of an enumerated parameter always returns the short form data in upper case.

Example:

```
                :SOURce:PRESsure:SLEW:MODE?
```

queries rate mode, reply:

```
                                MAX
```

Boolean Data
Boolean data can only be one of two conditions; the numbers 1 and 0.

Example:

```
        :OUTPut:STATe 1
```

A query of boolean data always returns 1 or 0.

String Data
String data can contain any of the ASCII characters. A string must start with a double "quote" (ASCII 34) or a single `quote` (ASCII 39) and end with the same character.

Note:
Characters in a string in either double "quote" or single `quote` are case sensitive.

Example:

```
        :SOURCe[:PRESsure]:RANGe '2BARG'
```

or

```
        :SOURCe[:PRESsure]:RANGe "2BARG"
```

selects the 2 bar g range.

A query of a string parameter always returns the string in double "quotes".

Intentionally left blank

3 STATUS SYSTEM

The status reporting system informs the external controller that an event has occurred. This information is in the form of a service request (SRQ) using IEEE 488 or an SRQ message using RS232.

The PACE Series uses status reporting as defined in IEEE 488.2 with the implementation of status registers.

The OPERation status registers have been implemented to comply with the SCPI protocol. These are registers where the individual bits are summary bits of the status of the instrument. Since the SCPI protocol does not include pressure instruments, bit 10 of both these registers are used as a pressure summary bit. This pressure summary bit is expanded to two, 16 bit registers (Bit 15 is not used and is always zero).

The only bit implemented in the Operation status register is bit 10 (summary of the pressure operation status).

A summary bit is the final output of a data structure, it is a single bit that shows the status of one or more related events in the instrument. The basic structure of a summary bit

- Condition register
- Event register
- Enable register
- Logical ANDing of the Event and Enable registers
- Summary bit that summarises the result using OR logic

Condition Register

This register shows the current status of the device. The condition register is constantly updated - the bits in the register are set or reset showing the current condition.

Event Register

The event register shows an event that occurs in the condition register (a condition bit goes from low to high). This condition change is stored and only reset when the event register is read or the *CLS command sent.

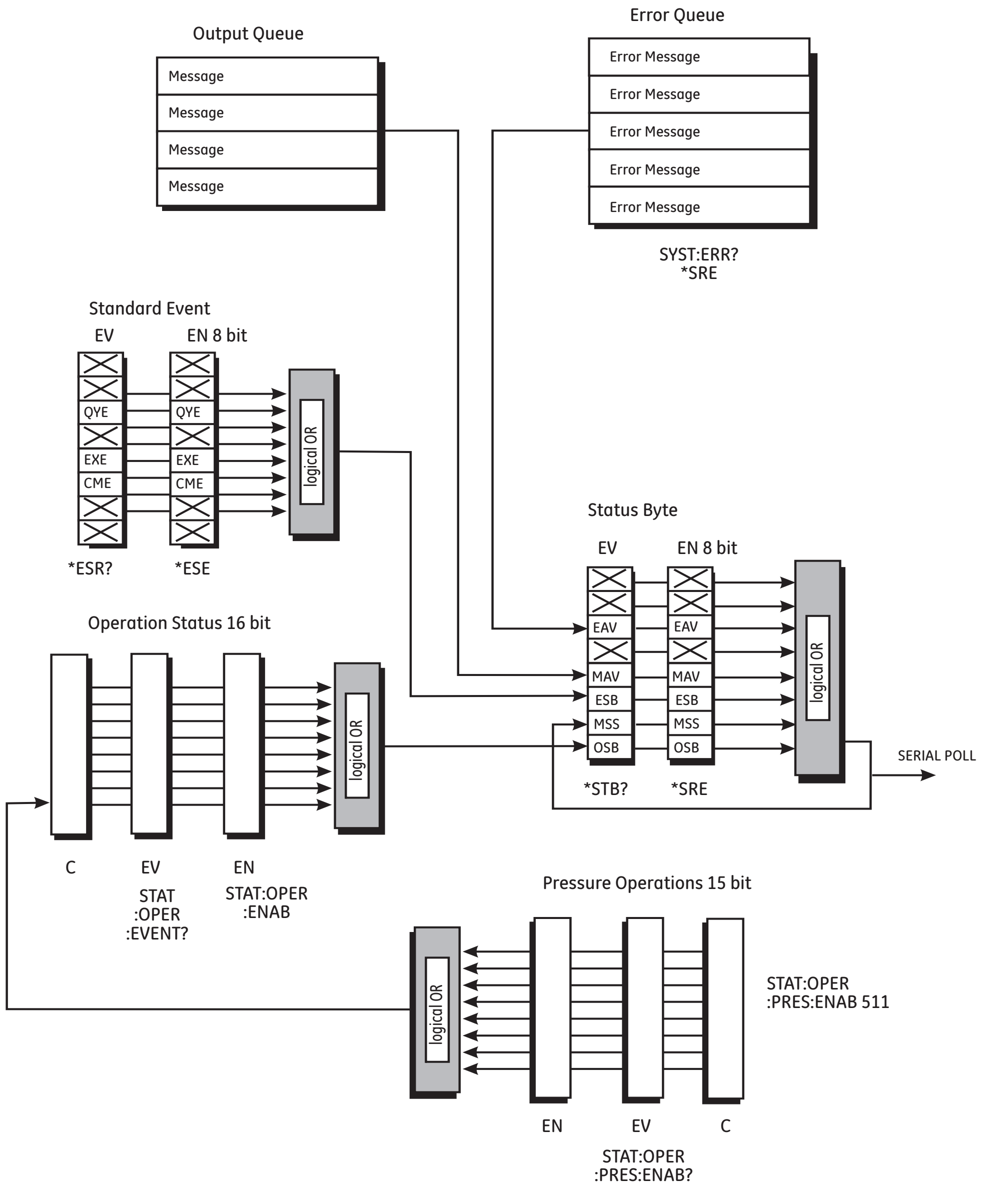
Enable Register

This register allows the results of the event register to pass through to the next cascaded register and enables the user to select the event that should generate the final SRQ event.

The status system implemented in the instrument is shown in the following diagram:

Note:

Initial values of registers are 0, with the queues empty.



Key:

C	=	*Condition - variable values*
EV	=	*Latched values*
EN	=	*Bit mask*

Figure 3-1 Status System

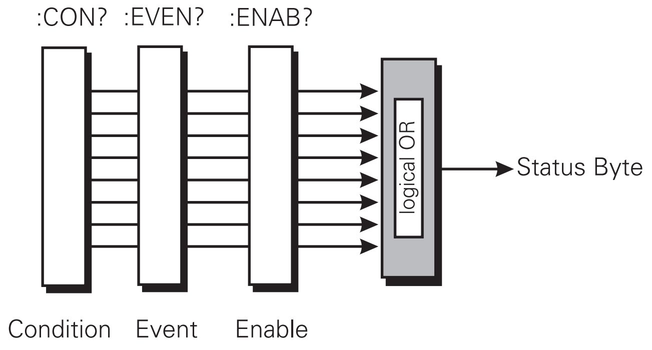
3.1 Output queue

The output queue is a text readable data queue that is read through the IEEE 488 talk command. The queue is cleared by reading all elements in it or by the *CLS command.

Every time a query has been successfully completed, the response, in a text readable format is placed at the end of the output queue. If the MAV bit in the "Status Byte" was previously cleared it will be set. The output queue can contain up to 256 characters. If there is not enough space in the output queue for a new message, the error -350, "Queue overflow" will be placed into the error queue and the most recent output message will be lost.

3.2 Standard event group

The standard event group are 8 bit registers that are read by the IEEE 488 standard commands. The event register is cleared by reading it; the event and enable registers are cleared by the *CLS command.



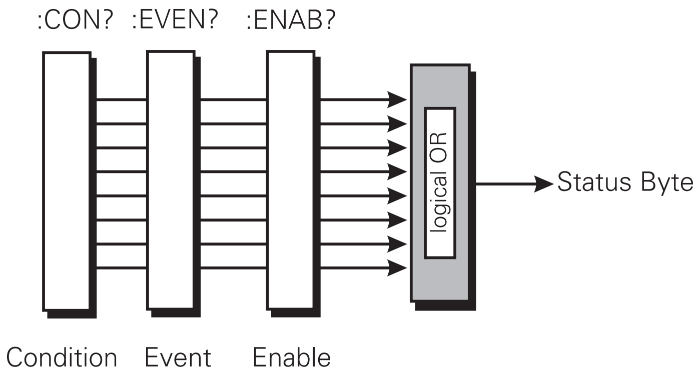
Bits within the standard event condition register are set by system errors and events. In addition to setting the status bits, a text message will be placed in the error queue. The ESB bit in the status byte sets if the associated bit in the event enable register is set. The enable register may be set through the *ESE command so that selected standard events cause the ESB bit to be set. The system events that set each bit are as follows:

Bit	Name	Description	Meaning/data
0	OPC	Not used	Reserved currently returns 0
1	RQC	Not used	Reserved currently returns 0
2	QYE	-400 to -499	Query errors
3	DDE	Not used	Reserved currently returns 0
4	EXE	-200 to -299	Execution errors
5	CME	-100 to -199	Command errors
6	URQ	Not used	Reserved currently returns 0
7	PON	Not used	Reserved currently returns 0

Table 3-1 Standard Event Register

3.3 Operation status group

The operation status group are 16 bit registers that are read by the STAT:OPER commands. The event register is cleared by reading it; the event and enable registers are cleared by the *CLS command.



When a standard operation condition occurs an appropriate bit is set in the condition register (this clears when the condition no longer exists). The bit is then latched in the event register. If the associated bit in the enable register is set, the OPR bit in the status byte sets. The enable register may be set through the STAT:OPER:ENAB command so that only selected standard operation events cause the OPR bit to set.
Problems can occur with some IEEE 488 controllers reading 16 bit unsigned numbers. All registers in this group do not use bit 15. The enable bit cannot be set and when read returns 0. The condition register is defined as follows:

Vent complete
This signal occurs when the controller has been requested to vent and the vent has completed or timed out.

Range change complete
This signal occurs when the controller has been requested to perform a range change and the range change is complete.

In-Limits reached
This signal is set every time the controlled pressure is within the specified limits. The signal is only generated if the pressure has been within limits for a user defined wait time period.

Zero complete
This signal is generated when a manual or timed zero is complete. If the zero times out then this signal is also generated.

Bit (1)	Data (2)	Bit (3)	Data (4)
0	Vent complete	1	Range change complete
2	In-limits reached	3	Zero complete
4	Auto-zero started	5	Fill time, timed-out
6	Reserved - returns 0	7	Reserved - returns 0
8	Switch contacts changed state	9	Reserved - returns 0
10	Reserved - returns 0	11	Reserved - returns 0
12	Reserved - returns 0	13	Reserved - returns 0
14	Reserved - returns 0	15	Reserved - returns 0

Table 3-2 Operation Status Register

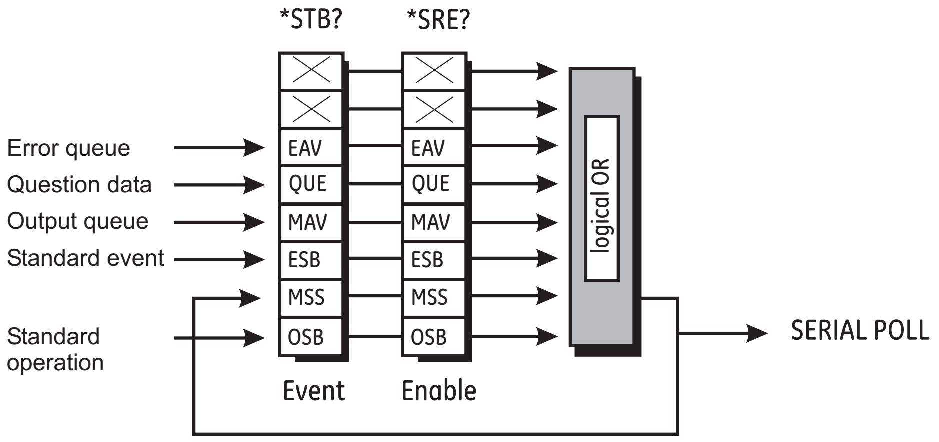
Auto zero started
When the controller is in the auto zero mode this signal indicates that the auto zero process has started. Thezero complete signal indicates that the zero process has finished.

Fill timed out
If a set-point has been requested and the set-point cannot be achieved within the fill timeout time, the fill timed out signal is generated.

Switch contacts changed state
Every time the switch contacts used for performing a switch test change state this bit is set.

3.4 Status Byte group

The status byte group are 8 bit registers that are read by the IEEE 488 standard commands. The event register is cleared by reading it; the event and enable registers are cleared by the *CLS command.



Bits within the status byte are a summary of other data structures in the status system. These bits will become set if other parts of the status system indicates that they should do so (i.e., a message in the output queue or error queue or, a condition and enable set in a register pair).
If the associated bit in the status enable register is set, a serial poll is generated and bit 6 is set. The enable register may be set through the *SRE command so that only selected status bits cause a serial poll.

Note: *Bit 6 of the enable register is always set to 0.*

There are some small differences between * STB? and serial polling. Either method can be used to read the state of bits 0-5 and bit 7. The reading method is different for bit 6 when using *STB? and serial poll. In general, use serial polling inside interrupt service routines, not *STB?

Bit 2 - EAV sets when there is an error in the error queue. The :SYST:ERR? command has to be sent to retrieve the error. The error queue buffers a maximum of five errors. When no more errors are available the message “No Error” is returned.

Bit 4 - MAV sets when there is a message available in the output queue.

Bit 5 - ESB sets when a standard event has occurred in the Standard Event Register.

Bit 6 - MSS sets when an SRQ is generated - SRQ sets when both the Status byte and the Service Request Enable register are at logic 1 (AND function).

RS232 Specific
A service request (SRQ) produces the message::SRQ <value>

where:
<value> = the contents of the status summary byte.

The status system data structure sets each bit as follows:

Bit	Name	Description
0	-	Reserved currently returns 0
1	-	Reserved currently returns 0
2	EAV	Error in errror queque
3	-	Reserved currently returns 0
4	MAV	Messages available in output queque
5	ESB	Summary bit from standard event
6	MSS	Summary bit after service request - SRQ
7	OSB	Summary bit from standard operations status

Table 3-3 Status Byte Register

Example commands using the Status Byte and Status Byte Enable registers:

***SRE 16**	*Generate an SRQ interrupt when messages are available.*
***SRE?**	*Find out what events are enabled to generate SRQ interrupts.*
***STB?**	*Read and clear the Status Byte Enable register.*

IEEE 488 Specific
Bit 7 - OSB sets when the pressure operations register bit 10 changes state. The operations register is a 16 bit register only using bit 10. This bit is a summary of the pressure operations register.

To set-up the status reporting system.

1 All status registers should be cleared by the command:
*CLS

2 The Pressure Operations Event register has to be set to enable the Pressure Operations Condition Register to send all the events to be reported; use the command:

:STAT:OPER:PRES:ENAB 511

The enabled events may also be read by the query:

:STAT:OPER:PRES:ENAB?

3 The Operation Status Event register must then be enabled to read bit 10 by the command:

:STAT:OPER:ENAB 1024

The enabled events may also be read by the query:

:STAT:OPER:ENAB?

4 The status request to enable the SRQ must then be set.

To enable only the Operation Status register (OSB) send the command:

*SRE 128

To enable the Operation Status register (OSB) and the Error Queue (EAV) send the command:

*SRE 132

This register may also be read by the query:

*SRE?

An event occurring generates an SRQ, the Status Byte should be queried to find the source of the event.

If bit 2 of the Status Byte Register is set the error queue can be read by the query:

```
:SYST:ERR?
```

Keep issuing this query until there are no more errors in the error queue. At this point, bit 2 of the Status Byte Register clears.

If bit 7 of the Status Byte Register is set the Pressure Operations event register can be read by the query:

```
:STAT:OPER:PRES?
```

returning the bits of events that have occurred. Reading this register clears it and the associated status bit (bit 7).

At any time the instantaneous status of the pressure system can be read by the query:

```
:STAT:OPER:PRES:COND?
```

3.5 Instrument Errors

Any instrument error that occurs, either programming errors or execution errors, is stored in an error queue which is separate from the main output queue. The errors can be read by issuing the following command query:

```
:SYST:ERR?
```

The error queue can hold up to five errors. Each time the error queue is queried the instrument responds with the next stored error in the queue. The response consists of an error number followed by a string describing the error. When the error queue is empty the instrument responds with:

```
0,"No error"
```

Querying the error queue clears the storage location in the error buffer. If more than five errors occur, before being queried, the 'Queue overflow;Error queue overflow' message is placed into the error queue. All subsequent errors are lost until the error queue is cleared.

4. COMMAND AND QUERY SUMMARY

The following lists of all the SCPI commands and queries that apply to the instrument.

4.1 Command structure

Some of the commands in the following summary are enabled at specific times and conditions, most can be enabled at any time. The command structure divides into subsystems as follows:

Command sub-system

:CALibration - calibration commands.

:INSTrument - instrument specific commands.

INPut - sets the switch input of the control module.

:OUTPut - controls the output pressure and logical outputs.

:SENSe - directs the instrument to measure selected parameters.

:SOURce - the commands that control the pressure outputs.

:STATus - instrument state.

:SYSTem - errors and SCPI version.

:UNIT - sets the units for the instrument.

Common SCPI commands - three letter commands, prefixed by *.

Instrument control commands - three letter commands, prefixed by **:**.

- This section describes each command in detail including parameters passed to it and response data returned. The general short form command is shown at the top of each page. The following information is then given:

Command Syntax	-	The upper case represents the short form command.
Parameter	-	Type: DECIMAL, INTEGER, ENUMERATED CHARACTER, BOOLEAN or STRING.
Short form	-	The short alternative for the command to be effective.
Function	-	Basic function of the command.
Default	-	The default value or the maximum and minimum values where appropriate.

Query Syntax	-	The upper case represents the short form query command.
Parameter	-	Type: DECIMAL, INTEGER, ENUMERATED CHARACTER, BOOLEAN or STRING.
Short form	-	The short alternative for the query to be effective.
Function	-	Basic function of query command.
Response	-	Data returned by the instrument following the query command.

- Description

  Details of the command and query with any conditions of use and any related commands.

  Note:

  Many of the command descriptions contain an example code: sent (Tx) to the instrument and the data received (Rx) from the instrument.

CALibration

The CALibration subsystem enables the calibration of the transducers and the rate control system, refer to the user manual for further details.

:CAL:PRES:POIN

Command Syntax

n/a

Parameter:

Short form:	
Function:	
Defaults:	

Query Syntax

:CALibration:[PRESsure]:POINts?

Short form:	CAL:POIN?
Function:	Gets the number of calibration points
Response:	3

Description

Valid only when calibration is enabled, this queries the number of calibration points.

:CAL:PRES:ACC

Command Syntax

:CALibration:[PRESsure]:ACCept

Parameter:	Integer 1
Short form:	:CAL:ACC
Function:	Accepts calibration values
Defaults:	no default value

Query Syntax

n/a

Short form:
Function:
Response:

Description
Valid only when calibration is enabled, this command accepts the calibration values entered in the calibration process.

:CAL:PRES:VAL

Command Syntax

:CALibration:[PRESsure]:VALue[x]

where: x = 1, 2 or 3

Parameter:	<decimal>
Short form:	:CAL:VAL
Function:	Enables calibration value to be entered.
Defaults:	

Query Syntax

:CALibration:[PRESsure]:VALue[x]?

Short form:	:CAL:VAL?
Function:	Queries calibration point value
Response:	Returns pressure value for VALue[x].

Description

Valid only when calibration is enabled, this command enables a calibration value to be entered during the calibration process.

x	3 Calibration points
1	lowest pressure
2	middle pressure
3	highest pressure

Example for a 2 bar unit

```
CAL:PRES:VAL1 -0.9
CAL:PRES:VAL2 0.0
CAL:PRES:VAL3 2.0
```

:CAL:ZERO:VALV

Caution: **Opening the zero valve with high pressure in the system can cause damage to the equipment. Reduce the system pressure and make sure the controller is OFF before opening the zero valve.**

Command Syntax
:CALibration:ZERO:VALVe

Parameter:	<boolean>
Short form:	:CAL:ZERO:VALV
Function:	Opens and closes zero valve.
Default:	0

Query Syntax
:CALibration:ZERO:VALVe?

Short form:	CAL:PRES:ZERO:VALV?		
Function:	Queries state of valve.		
Response:	1	-	open
	0	-	close

Description

This command is used to open and close the zero valve. The query gets the state of the zero valve - open or close.

Example

```
TX> :CAL:PRES:ZERO:VALV 1
TX> :CAL:PRES:ZERO:VALV:STAT 1
```
Either of above two commands opens the zero valve.

```
Either:   TX> :CAL:PRES:ZERO:VALV?
           RX> :CAL:PRES:ZERO:VALV 1
Or         TX> :CAL:PRES:ZERO:VALV:STAT?
           RX> :CAL:PRES:ZERO:VALV:STAT 1
```

```
TX> :CAL:PRES:ZERO:VALV: 0
TX> :CAL:PRES:ZERO:VALV:STAT 0
```
Either of above two commands close the zero valve, switch OFF the controller.

```
Either:   TX> :CAL:PRES:ZERO:VALV?
           RX> :CAL:PRES:ZERO:VALV 0
Or         TX> :CAL:PRES:ZERO:VALV:STAT?
           RX> :CAL:PRES:ZERO:VALV:STAT 0
```
Gets the condition of the zero valve.

:CAL:ZERO:VALV:STAT

Command Syntax

:CALibration:ZERO:VALVe:STATe

Parameter:	<boolean>
Short form:	:CAL:ZERO:VALV:STAT
Function:	Opens and closes zero valve.
Default:	0

Query Syntax

:CALibration:ZERO:VALVe:STATe?

Short form:			
Function:	Queries state of valve.		
Response:	1	-	open
	0	-	close

Description

This command is used to open and close the zero valve. The query gets the state of the zero valve - open or close.

:CAL:ZERO:AUTO

Command Syntax

:CALibration:ZERO:AUTO <Boolean>

Parameter:	<boolean>			
		0	-	aborts a zero process
		1	-	starts a zero process

Short form:	:CAL:ZERO:AUTO
Function:	Pressure zeroing
Default:	0

Query Syntax

:CALibration:ZERO:AUTO?

Short form:	:CAL:ZERO:AUTO?
Function:	Query progress of zero
Response:	
	0 - Zero OK
	1 - Zero in progress

Description

This command starts or aborts a zero process. The progress of the zero can be monitored by using the query.

INPut

The INPut subsystem shows the state of the logical inputs.

:INP:LOG

Command Syntax

n/a

Parameter:

Short form:
Function:
Default:

Query Syntax

:INPut:LOGic?

Short form:	:INP:LOG?		
Function:	Asks for state of switch input within the controller module.		
Response:	first parameter	-	0 = OFF, 1 = ON
	second parameter	-	measured pressure at the time of switching (snapshot in current pressure units).

Description
This command queries the state of the switch input within the module and the pressure at time of switching operations.

Example
:INP:LOG?
:INP:LOG 0, 0.8321209
Current logic OFF, pressure was 0.8321209 in current pressure units when the module was switched to OFF condition.

INSTrument

The INSTrument subsystem gets information about the configuration of the instrument .

:INST:CAT

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:INSTrument:CATalog?

Short form:	:INST:CAT?
Function:	Query ranges fitted
Response:	A list of comma-separated strings of ranges fitted.

Description
This query returns a list of ranges fitted to the instrument. The reply is a comma separated list of strings representing each range.

e.g. “2barg”,”3.5bara”.

If a barometer is fitted, the string “BAROMETER” is added to the list.
Example

```
TX> :INST:CAT?
RX> :INST:CAT "3.50barg","BAROMETER","4.50bara"
```

:INST:CAT:ALL

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:INSTrument:CATalog:ALL?

Short form:	:INST:CAT:ALL?
Function:	Query all ranges fitted
Response:	A list of comma separated strings of ranges fitted.

Description
This query returns all the ranges fitted to the instrument. The reply is a comma separated list of strings representing each range.

Example

```
TX> :INST:CAT:ALL?
RX> :INST:CAT:ALL "3.50barg","BAROMETER","4.50bara"
```

:INST:LIM

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:INSTrument:[LIMits][x]?

Short form:	:INST?
Function:	Queries the upper and lower full-scale limits of the fitted sensor ranges. The index number x is used to index into the list of available ranges. *example:*

x	sensor
1	control sensor
2	source pressure +ve
3	source pressure -ve
4	barometric range (optional)

Response:	A string representing the range, a number representing the upper full-scale and a number representing the lower full-scale.

Description

This query returns the name of the range as a string and the upper and lower pressure limits in the selected units:

<u>Example</u>

INST:LIM? X=1, 2, 3, and 4

```
TX> :INST:LIM?
RX> :INST:LIM "3.50barg", 3675.0000000, -
1100.0000000
```

```
TX> :INST:LIM1?
RX> :INST:LIM "3.50barg", 3675.0000000, -
1100.0000000
```
Returns the control sensor limits.

```
TX> :INST:LIM2?
RX> :INST:LIM2 "10.00 barg", 10500.0000000, -
1100.0000000
```
Returns the +ve source sensor limits.

```
TX> :INST:LIM3?
RX> :INST:LIM3 "1.00 barg", 1050.0000000, -
1100.0000000
```
Returns the –ve source sensor limits.

```
TX> :INST:LIM4?
RX> :INST:LIM4 "BAROMETER", 1207.5000000,
825.0000000
```
Returns the barometric sensor limits.

INST:SENS:CALD

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:INST:SENS[X]:CALD[Y]?

Short form: :INST:SENS:CALD?
Function: Queries sensor calibration dates.
Response: Returns sensor [x] calibration date:
where x is the sensor.
X = 1, 2, 3 and 4.

x	sensor
1	control sensor
2	source pressure +ve
3	source pressure -ve
4	barometric range (optional)

where y is the yth calibration date for sensor x.
y = 1, 2, 3 . . . . .10 calibration dates stored for each sensor.

Description

This command returns the following:

```
TX> :INST:SENS:CALD?
RX> :INST:SENS:CALD 2008, 10, 25
```

```
TX> :INST:SENS1:CALD?
RX> :INST:SENS1:CALD 2008, 10, 25
```
Returns the control sensor calibration date.

```
TX> :INST:SENS2:CALD?
RX> :INST:SENS2:CALD 2008, 5, 21
```
Returns the +ve source sensor calibration date.

```
TX> :INST:SENS3:CALD?
RX> :INST:SENS3:CALD 2008, 11, 1
```
Returns the -ve source sensor calibration date.

```
TX> :INST:SENS4:CALD?
RX> :INST:SENS4:CALD 2009, 2, 6
```
Returns the barometric sensor calibration date.

:INST:SENS:FULL

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:INST:SENS[X]:FULL?

Short form:	:INST:SENS:FULL?
Function:	Queries sensor full-scale value.
Response:	Returns the full-scale value of the selected sensor.

x	sensor full-scale
1	control sensor full-scale
2	+ve source sensor full-scale
3	-ve source sensor full-scale
4	barometric sensor full-scale (optional)

Description

This query returns the following:
X=1, 2, 3 and 4

```
:INST:SENS:FULL?
:INST:SENS:FULL 3.5000000
```

```
:INST:SENS1:FULL?
:INST:SENS:FULL 3.5000000
```
Returns the control sensor full-scale.

```
:INST:SENS2:FULL?
:INST:SENS2:FULL 10.0000000
```
Returns the +Ve source sensor full-scale.

```
:INST:SENS3:FULL?
:INST:SENS3:FULL 1.0000000
```
Returns the -Ve source sensor full-scale.

```
:INST:SENS4:FULL?
:INST:SENS4:FULL 1.1500000
```
Returns the barometric sensor full-scale.

:INST:SN

Command Syntax

:INStrument:SN

Parameter:	<integer>
Short form:	:INST:SN
Function:	Assigns a serial number to the instrument

Query Syntax

:INStrument:SN?

Short form:	:INST:SN?
Function:	Used to query the serial number of the instrument. Asks for serial number.
Response:	Integer representing serial number.

Description

This query returns the serial number of the instrument.

Example

```
TX> :INST:SN?
RX> :INST:SN 58784
```

Returns the instrument serial number.

:INST:VERS

Command Syntax

n/a

Parameter:
Short form:
Function:

Query Syntax

:INSTrument:VERSion[x]?

Short form:	:INST:VERS[x]?
Function:	Queries the software versions of the controller.
Response:	Returns strings representing the software version:

x	software
1	instrument main code
2	instrument OS build
3	instrument boot ROM
4	module main code
5	module boot ROM

Description

<u>Example</u>
:INST:VERS[X]? X=1, 2, 3, 4 and 5

```
TX> :INST:VERS?
RX> :INST:VERS "01.05.02"
```

```
TX> :INST:VERS1?
RX> :INST:VERS "01.05.02"
```
Software version for instrument main code.

```
TX> :INST:VERS2?
RX> :INST:VERS2 "01.06.03"
```
Software version for instrument OS build.

```
TX> :INST:VERS3?
RX> :INST:VERS3 "01.00.00"
```
Software version for instrument boot ROM.

```
TX> :INST:VERS4?
RX> :INST:VERS4 "01.03.39"
```
Software version for module main code.

```
TX> :INST:VERS5?
RX> :INST:VERS5 "01.00.00"
```
Software version for module boot ROM.

OUTPut

The OUTPut subsystem turns the pressure controller on/off and controls the state of the logical outputs.

:OUTP:STAT

Command Syntax

:OUTPut:STATe <Boolean>

Parameter:	<boolean>			
		0	-	turn controller off
		1	-	turns controller on

Short form:	:OUTP
Function:	Turn the pressure controller on/off
Default:	0

Query Syntax

:OUTPut:STATe?

Short form:	:OUTP?
Function:	Asks for state of pressure controller
Response:	0 - controller off
	1 - controller on

Description
Sets or queries the state of the pressure controller.

Example

```
TX> :OUTP:STAT?
RX> :OUTP:STAT 0
```

The controller currently turned off.

To turn the controller on:

```
TX> :OUTP:STAT ON
TX> :OUTP:STAT?
RX> :OUTP:STAT 1
```

To turn the controller off:

```
TX> :OUTP:STAT OFF
```

:OUTP:LOG

Command Syntax

:OUTP:LOGic[x]

Parameter:	<boolean>
	OFF - turn relay [x] OFF
	ON - turn relay [x] ON
	[x] = 1, 2 or 3
Short form:	:OUTP:LOG
Function:	Turns relay ON and OFF
Default:	Relay 1

Query Syntax

:OUTP:LOGic[x]?

Short form:	:OUTP:LOG[x]?
Function:	Asks for relay [1,2,3] condition.
Response:	0 - relay OFF
	1 - relay ON

Description
With the volt-free contact option installed. The three relays can be queried and switched on and off, where [x] = 1, 2, 3 represents the relay number. If x is omitted, then x = 1.
These commands return an error message: "VOLT-FREE CONTACT option not installed", if the volt-free contact option is not installed.
Example
:INST:LOG[X]? X=1, 2 and 3

```
TX> :OUTP: LOGic?
RX> :OUTP:LOGic 0   //relay1 currently OFF

TX> :OUTP:LOGic ON
TX> :OUTP:LOGic?
RX> :OUTP:LOGic 1  //relay1 ON

TX> :OUTP: LOGic2?
RX> :OUTP:LOGic2 0   //relay2 currently OFF

TX> :OUTP:LOGic2 ON
TX> :OUTP:LOGic2?
RX> :OUTP:LOGic2 1   //relay2 ON

TX> :OUTP: LOGic3?
RX> :OUTP:LOGic3 0   //relay3 currently OFF

TX> :OUTP:LOGIc3 ON
TX> :OUTP:LOGIc3?
RX> :OUTP:LOGIc3 1   //relay3 ON
```

SENSe

The SENSe subsystem selects, configures and queries the measurement functions of the instrument.

:SENS:PRES

Command Syntax

n/a

Parameter:
Short form:
Function:

Query Syntax

:SENSe[:PRESsure]?

Short form:	:SENS?
Function:	This query reads the sensor which has been selected by the RANGE command.
Response:	A decimal, pressure reading in the current units.

Description

Queries the pressure reading for the selected sensor in the selected units. The sensor can be changed see, :SENSe[:PRESsure]:RANGe and :SOURe[:PRESsure]:RANGe commands.

:SENS:PRES:INL

Command Syntax

n/a

Parameter:
Short form:
Function:

Query Syntax

:SENSe[:PRESsure]:INLimits?

Short form:	:SENS:INL?			
Function:	Query in-limits value			
Response:	First parameter - current pressure.			
	Second parameter - in limit:			
		0	=	not in limits
		1	=	in limits

Description
This command query gets the in-limits value and the in-limits status:

<u>Example</u>

```
TX> :SENS:PRES:INL?
RX> :SENS:PRES:INL 990.0527344, 0

TX> :SENS:PRES:INL?
RX> :SENS:PRES:INL 990.0527344, 1
```

‘0’ = not yet in limit;
‘1’ = already in limit;

:SENS:PRES:SLEW

Command Syntax

n/a

Parameter:	
Short form:	
Function:	

Query Syntax

:SENSe[:PRESsure]:SLEW?

Short form:	:SENS:slew?
Function:	Asks for current slew rate.
Response:	Decimal number representing slew rate in current pressure units per second.

Description
This query gets the slew rate of the input pressure. A constant input pressure gives a slew rate of: 0.0.

:SENS:PRES:BAR

Command Syntax

n/a

Parameter:
Short form:
Function:

Query Syntax

:SENSe[:PRESsure]:BARometer?

Short form:	:SENS:BAR?
Function:	Queries the barometric pressure value.
Response:	<number> in the selected units of pressure measurement.

Description
Returns the barometric pressure measured by the optional barometric transducer. If the optional barometric transducer is not fitted the response is zero pressure.

Example
:SENS:PRES:BAR?
:SENS:PRES:BAR 982.8430904 (this value in mbar).

:SENS:PRES:RANG

Command Syntax

:SENSe[:PRESsure]:RANGe <string>

Parameter:	<string>	range information

Short form: :SENS:RANG
Function: Used to select a range to be sensed.

Query Syntax

:SENSe[:PRESsure]:RANGe?

Short form: :SENS:RANG?
Function: Asks for currently sensed range.
Response: String representing selected pressure range.

Description

This command selects the pressure range to be used for returning the pressure reading, also see the SOURCe:RANGe command.

Example

Ranges of the instrument can be found by the query:

```
TX> :INST:CAT:ALL?
RX> :INST:CAT:ALL "3.50barg","BAROMETER","4.50bara"
```

This shows the instrument has three ranges.

The current range selected can be found by the query:

```
TX> :SENS:PRES:RANG?
RX> :SENS:PRES:RANG "3.50barg"
```

To select a different range by the command:

```
TX> :SENS:PRES:RANG "4.50bara"
```

Selection confirmed by query:

```
TX> :SENS:PRES:RANG?
RX> :SENS:PRES:RANG "4.50bara"
```

Note:
The command parameter, "4.50bara", has to be typed exactly. It is case-sensitive and not typo-error tolerant. The command does not affect on instrument front-panel display. Use :SOUR:PRES:RANG to change front-panel range display. (See instrument user's manual).

:SENS:PRES:RES

Command Syntax

:SENSe[:PRESsure]:RESolution <string>

Parameter:	<integer>	resolution information

Short form:	:SENS:RES
Function:	Used to select a resolution to be used.

Query Syntax

:SENSe[:PRESsure]:RESolution?

Short form:	:SENS:RES?
Function:	Asks for current resolution.
Response:	An integer from 4 to 7 representing selected resolution.

Description
This command selects the resolution to be used for showing the pressure reading in the front panel display.
Example

```
TX> :SENS:PRES:RES?
RX> :SENS:PRES:RES 6
```
The resolution can be changed by the command, such as:
```
TX> :SENS:PRES:RES 4
```
Query again confirms that the resolution has been changed to ‘4’ :
```
TX> :SENS:PRES:RES?
RX> :SENS:PRES:RES 4
```
Change the resolution back to default by sending:
```
TX> :SENS:PRES:RES 6
```
The parameter for this command is an integer from 0 to 6.
Sending:
```
TX> :SENS:PRES:RES 7
```
Causes an error:
```
TX> :SYST:ERR?
RX> :SYST:ERR -222,"Data out of range; Parameter 1"
```

:SENS:PRES:CORR:HEAD

Command Syntax

:SENSe[:PRESsure]:CORRection:HEAD <enumerated>,<decimal>

Parameters:	<enumerated>	AIR	-	Air used as gas
		NITRogen	-	Nitrogen used as gas
	<numeric>	Height of gas in metres.		

Short form::	SENS:CORR:HEAD <enumerated>,<numeric>	
Function:	Head correction parameters	
Default:	Enumerated	AIR
	decimal	0

Query Syntax

:SENSe[:PRESsure]:CORRection:HEAD?

Short form:	SENS:CORR:HEAD?
Function:	Query gas and height of head correction
Response:	AIR/NITRogen and height in metres (+100 to -100)

Description

A correction must be made if the unit under test is at a different height from the instrument. This command programs the gas used and the height difference.

Example

```
TX> :SENS:PRES:CORR:HEAD?
RX> :SENS:PRES:CORR:HEAD AIR, -0.7500000
```

Head correction can be set to a new height by the command:

```
TX> :SENS:PRES:CORR:HEAD AIR, 1.2
```

Another query will confirm the height change:

```
TX> :SENS:PRES:CORR:HEAD?
RX> :SENS:PRES:CORR:HEAD AIR, 1.2000000
```

Head correction can be set for another gas by:

```
TX> :SENS:PRES:CORR:HEAD NITROGEN, 1.2
```

Another query will confirm the height change:

```
TX> :SENS:PRES:CORR:HEAD?
RX> :SENS:PRES:CORR:HEAD NITR, 1.2000000
```

Note: *NITROGEN or NITR is an enumerated data type (not in punctuation marks), not a string and is not case-sensitive and can have a short form.*

:SENS:PRES:CORR:HEAD:STAT

Command Syntax

:SENSe[:PRESsure]:CORRection:HEAD:STATe <Boolean>

Parameter	<boolean>			
		0	-	Disables head correction
		1	-	Enables head correction

Short form::	SENS:CORR:HEAD:STAT <Boolean>
Function:	Enables / disables head correction.
Default:	0

Query Syntax

:SENSe[:PRESsure]:CORRection:HEAD:STATe?

Short form:	:SENS:CORR:HEAD:STAT?
Function:	Query head correction state
Response:	0 - head correction off
	1 - head correction on

Description

This command enables or disables the head correction compensation.

Example

```
TX> :SENS:PRES:CORR:HEAD:STATe?
RX> :SENS:PRES:CORR:HEAD:STAT 0
Head correction off
It can be turned on and off by:
TX> :SENS:PRES:CORR:HEAD:STATe on
TX> :SENS:PRES:CORR:HEAD:STATe off
Or by:
TX> :SENS:PRES:CORR:HEAD:STATe 1
TX> :SENS:PRES:CORR:HEAD:STATe 0
```

:SENS:PRES:CORR:OFFS

Command Syntax

:SENSe[:PRESsure]:CORRection:OFFSet

Parameter:	<decimal> tare offset value in current pressure units.
Short form:	:SENS:OFFS
Function:	Subtracts the offset value from the processed reading.
Default:	0

Query Syntax

:SENSe[:PRESsure]:CORRection:OFFSet?

Short form:	:SENS:OFFS?
Function:	Asks for the tare value.
Response:	Number corresponding to the tare offset value.

Description

Example

```
TX> :SENS:PRES:CORR:OFFS?
RX> :SENS:PRES:CORR:OFFS 0.0
```

Offset is zero.

Offset can be changed to 100 mbar by sending, (instrument must be in mbar):

```
TX> :SENS:PRES:CORR:OFFS 100
```

Send query again to confirm offset has changed:

```
TX> :SENS:PRES:CORR:OFFS?
RX> :SENS:PRES:CORR:OFFS 100.0000000
```

Note:

The offset depends on the units of measurement used by the instrument. If changed from mbar to bar, the same query will be returned:

```
TX> :SENS:PRES:CORR:OFFS?
RX> :SENS:PRES:CORR:OFFS 0.1000000
```

:SENS:PRES:CORR:OFFS:STAT

Command Syntax

:SENSe[:PRESsure]:CORRection:OFFSet:STATe

Parameter:	<boolean>			
		0	-	disables offset
		1	-	enables offset

Short form: :SENS:OFFS:STAT
Function: Enables and disables the offset function.

Query Syntax

:SENSe[:PRESsure]:CORRection:HEAD:OFFSet:STATe?

Short form: :SENS:OFFS:STAT?
Function: Asks if offset function is on or off.
Response: 1 (on)
0 (off)

Description

This command enables and disables the offset function. The query gets the state of the offset (or tare) on or off.

Example

```
:SENS:PRES:CORR:OFFS:STATe
TX> :SENS:PRES:CORR:OFFS:STATe?
RX> :SENS:PRES:CORR:OFFS:STAT 0
The offset correction disabled.
Offset can be enabled and disabled by sending:
TX> :SENS:PRES:CORR:OFFS:STATe on
TX> :SENS:PRES:CORR:OFFS:STATe off
Or by:
TX> :SENS:PRES:CORR:OFFS:STATe 1
TX> :SENS:PRES:CORR:OFFS:STATe 0
```

:SENS:PRES:CORR:VOL

Command Syntax

n/a

Parameter:

Short form:	
Function:	
Default:	

Query Syntax

:SENSe[:PRESsure]:CORRection:VOLume?

Short form:	:SENS:PRES:CORR:VOL?
Function:	Ask for the estimated volume of the system connected to the instrument.
Response:	Decimal number in litres corresponding to volume.

Description
The instrument calculates the volume of the system connected by the amount of effort needed to attain a set-point.

<u>Example</u>

```
TX> :SENS:PRES:CORR:VOL?
RX> :SENS:PRES:CORR:VOL 0.0150000
```

:SENS:PRES:FILT:LPAS:BAND

Command Syntax

:SENSe[:PRESsure]:FILTer:[LPASs]:BAND <number>

Parameter:	<decimal> filter band response value in % full-scale.	
Short form:	:SENS:FILT:BAND	
Function:	Used to set-up the response band component of the filter.	
Default:	0	
	minimum	0
	maximum	100.0

Query Syntax

:SENSe[:PRESsure]:FILTer:[LPASs]:BAND?

Short form:	:SENS:FILT:BAND?
Function:	Ask for filter step response band parameter.
Response:	Number corresponding to filter step response value in % full-scale.

Description

The digital low pass filter has a response band configured as percentage of full-scale. e.g., defaults to 0.05 %FS. If the reading has changed by more than the configured band response value then the filtering is ignored for that conversion and the pressure goes instantly to the new value.

Example

:SENS:PRES:FILT:LPAS:BAND

```
TX> :SENS:PRES:FILT:LPAS:BAND?
RX> :SENS:PRES:FILT:LPAS:BAND 50.0000000
```

The current setting for the filter band is 50% of full-scale, i.e., the filter applies only when the change of pressure is within this band.

It can be changed to another number, (for example: 12% of full-scale), by sending

```
TX> :SENS:PRES:FILT:LPAS:BAND 12
TX> :SENS:PRES:FILT:LPAS:BAND?
RX> :SENS:PRES:FILT:LPAS:BAND 12.0000000
```

:SENS:PRES:FILT:LPAS:FREQ

Command Syntax

:SENSe[:PRESsure]:FILTer:[LPASs]:FREQuency <number>

Parameter:	<decimal> filter averaging time in seconds.
Short form:	:SENS:FILT:FREQ
Function:	Used to set up the averaging component of the filter.
Default:	0
	minimum 0
	maximum 20

Query Syntax

:SENSe[:PRESsure]:FILTer:[LPASs]:FREQuency?

Short form:	:SENS:FILT:FREQ?
Function:	Ask for filter average parameter.
Response:	Decimal number corresponding to filter average time in seconds.

Description

A digital low pass filter can be applied to the pressure reading. This is a first order low pass filter, the time constant depends on the value set by this command.

Note:

The decimal number used by the command and query does not represent frequency even though 'FREQ' is used.

Example

:SENS:PRES:FILT? TX> :SENS:PRES:FILT? RX> :SENS:PRES:FILT:LPAS:STAT 0	It can be set to another value, such as: TX> :SENS:PRES:FILT:LPAS:FREQ 1.76 TX> :SENS:PRES:FILT:LPAS:FREQ? RX> :SENS:PRES:FILT:LPAS:FREQ 1.7600000
:SENS:PRES:FILT:LPAS:FREQ TX> :SENS:PRES:FILT:LPAS:FREQ? RX> :SENS:PRES:FILT:LPAS:FREQ 2.0000000 The current setting for low-pass filter's time constant is 2 seconds.	

:SENS:PRES:FILT:LPAS:STAT

Command Syntax

:SENSe[:PRESsure]:FILTer[:LPASs]:[STATE] <Boolean>

Parameter:	<Boolean>		
	0	-	Disables low pass filter
	1	-	Enables low pass filter

Short form:	:SENS:FILT <Boolean>
Function:	Sets low pass filter ON or OFF.
Default:	OFF

Query Syntax

:SENSe[:PRESsure]:FILTer[:LPASs]:[STATe]?

Short form:	:SENS:FILT?
Function:	Query state (on or off) for the low pass filter
Response:	1 (ON) 0 (OFF)

Description

This command is used to enable or disable the low pass filter for producing a more stable reading. An 'intelligent' filter is implemented so that any noise in the system is filtered while step changes pass straight though the filter.

```
:SENS:PRES:FILT:LPAS:STAT
TX> :SENS:PRES:FILT:LPAS:STAT?
RX> :SENS:PRES:FILT:LPAS:STAT 0
```
The filter is currently off.

It can be set on and off by
```
TX> :SENS:PRES:FILT:LPAS:STAT on
TX> :SENS:PRES:FILT:LPAS:STAT off
```

Or by
```
TX> :SENS:PRES:FILT:LPAS:STAT 1
TX> :SENS:PRES:FILT:LPAS:STAT 0
```

SOURce

The SOURce subsystem controls the pressure output of the instrument.

:SOUR:PRES:COMP

Command Syntax

n/a

Parameter:

Short form:
Function:

Default:

Query Syntax

:SOURce[:PRESsure]:COMPensate [x]?

Short form:	:SOUR:COMP?
Function:	Queries +ve and -ve source pressures.
Response:	Pressure value in current pressure units.

Description
This query gets the +ve and -ve source pressures in the current pressure units.

<u>Example</u>

```
:SOUR:PRES:COMP[x]?
TX> :SOUR:PRES:COMP?
RX> :SOUR:PRES:COMP 3165.9526002
TX> :SOUR:PRES:COMP1?
RX> :SOUR:PRES:COMP 3165.9484591
TX> :SOUR:PRES:COMP2?
RX> :SOUR:PRES:COMP2 -963.9638062
```

x=1,2;

x	Source pressure measurements
1	+ve
2	-ve

If x is greater than 2, an error will be reported

```
TX> :SOUR:PRES:COMP3?
TX> :SYST:ERR?
RX> :SYST:ERR -114,"Header suffix out of range"
```

:SOUR:PRES:EFF

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:SOURce[:PRESsure]:EFFort?

Short form:	:SOUR:EFF?
Function:	This query only command returns the effort needed for the controller to achieve the set-point.
Response:	Decimal percent number representing controller effort.

Description
This query only command returns the % effort the controller does to achieve the set-point.

<u>Example</u>

```
TX> :SOUR:PRES:EFF?
RX> :SOUR:PRES:EFF -0.2342882
```

Percentage of effort the supply or vacuum valve makes to maintain the control point. The return number should be –100 to +100. A positive number indicates the supply valve makes more effort and a minus number indicates the vacuum valve makes more effort. If the controller is off:

```
TX> :OUTP off
TX> :SOUR:PRES:EFF?
RX> :SOUR:PRES:EFF 0.0
```

The return number is '0.0', since no valve is making any effort.

:SOUR:PRES:INL

Command Syntax

:SOURce[:PRESsure]:INLimits <number>

Parameter:	<decimal> in limits value as % full-scale
Short form:	:SOUR:INL <number>
Function:	The controller has an in-limits set-point indicator. This can generate a service request when the pressure is within limits for a set time period.
Default:	0.01 % full-scale
	minimum 0 % full-scale
	maximum 100 % full-scale

Query Syntax

:SOURce[:PRESsure]:INLimits?

Short form:	:SOUR:INL?
Function:	Query in-limits value
Response:	Decimal representing in-limits value as % full-scale.

Description

Sets the in-limits value.

Example

```
TX> :SOUR:PRES:INL?
RX> :SOUR:PRES:INL 0.0200000
```

The current in-limits set-point is 0.02 % full-scale

Can be set to:

```
TX> :SOUR:PRES:INL 0.01
TX> :SOUR:PRES:INL?
RX> :SOUR:PRES:INL 0.0100000
```

:SOUR:PRES:INL:TIME

Command Syntax

:SOURce[:PRESsure]:INLimits:TIME <number>

Parameter:	<integer>	in-limits time in seconds

Short form: :SOUR:INL:TIME <number>
Function: Sets the time that the pressure has to be within limits before generating a service request.

Default:	2 seconds	
	minimum	2 seconds
	maximum	999 seconds

Query Syntax

:SOURce[:PRESsure]:INLimits:TIME?

Short form: :SOUR:INL:TIME?
Function: Query in-limits timers.
Response: Number representing in-limits time in seconds.

Description
Sets the in-limits timer value.
Example

```
TX> :SOUR:PRES:INL:TIME?
RX> :SOUR:PRES:INL:TIME 2
```

Can be set to 99 by issuing command:

```
TX> :SOUR:PRES:INL:TIME 99
TX> :SOUR:PRES:INL:TIME?
RX> :SOUR:PRES:INL:TIME 99
```

Or to 999 by issuing command:

```
TX> :SOUR:PRES:INL:TIME 999
TX> :SOUR:PRES:INL:TIME?
RX> :SOUR:PRES:INL:TIME 999
```

:SOUR:PRES:LEV:IMM:AMPL

Command Syntax

:SOURce[:PRESsure][:LEVel][:IMMediate][:AMPLitude] <number>

Parameter:	<decimal> Pressure in current units
Short form:	SOUR <number>
Function:	Set the pressure set-point
Default:	0.0

Query Syntax

:SOURce[:PRESsure][:LEVel][:IMMediate][:AMPLitude]?

Short form	:SOUR?
Function:	Programmable set-point value
Response:	Decimal number representing pressure set-point in current units.

Description

This command sets the pressure set-point and is the long form of :SOUR:PRES.

<u>Example</u>

```
TX> :SOUR:PRES?
RX> :SOUR:PRES:LEV:IMM:AMPL 0.4000000
```

The current controller set-point is 0.4 of the current unit.

It can be set to another value, such as:

```
TX> :SOUR:PRES 0.5
TX> :SOUR:PRES?
RX> :SOUR:PRES:LEV:IMM:AMPL 0.5000000
```

:SOUR:PRES:LEV:IMM:AMPL:VENT

Command Syntax

:SOURce[:PRESsure][:LEVel][:IMMediate][:AMPLitude]:VENT <number>

Parameter:	<integer> 0 - abort vent 1 - start vent
Short form:	SOUR:VENT <integer>
Function:	Vents the user system.
Default:	0

Query Syntax

:SOURce[:PRESsure][:LEVel][:IMMediate][:AMPLitude]:VENT?

Short form:	:SOUR:VENT?
Function:	Query status of vent.
Response:	0 - vent OK 1 - vent in progress 2 - vent completed

Description

This command vents the user system; the command should be queried to get the status of the vent.

Examples

For an instrument switched on without any vent, a query:

```
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 0 (a return of ‘0’ indicates vent not in progress).
```

A vent can be started by sending:

```
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT 1
```

Immediate query:

```
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 1
```

A return of '1' indicates vent in progress

```
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 1
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 1
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 1
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 1
```

Continuing queries return '1', until the vent finishes and a return of '2':

```
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 2
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 2
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 2
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT?
RX> :SOUR:PRES:LEV:IMM:AMPL:VENT 2
```

To abort the vent process, send:

```
TX> :SOUR:PRES:LEV:IMM:AMPL:VENT 0
```

:SOUR:PRES:RANG

Command Syntax

:SOURCe[:PRESsure]:RANGe <string>

Parameter:	<string>	range information

Short form: :SOUR:RANG
Function: Selects the control range.

Query Syntax

:SOURCe[:PRESsure]:RANGe?

Short form:: SOUR:RANGE?
Function: Asks for currently selected controller range.
Response: String representing selected sense range.

Description
This command selects the range to be used for controlling pressure.
e.g. : SOURe:RANGe “2.0bara”
selects the 2 bar absolute range; the pressure units are always in bar.

Note:
Absolute ranges are pseudo-absolute values, combining barometric and gauge sensor readings.
<u>Example</u>
```
TX> :SOUR:PRES:RANG?
RX> :SOUR:PRES:RANG "3.50barg"
```
Queries current range

Can set to another range:
```
TX> :SOUR:PRES:RANG "4.50bara"
TX> :SOUR:PRES:RANG?
RX> :SOUR:PRES:RANG "4.50bara"
```

This change can be observed on front panel display.
The parameter is case sensitive.

:SOUR:PRES:SLEW

Command Syntax

:SOURCE[:PRESsure]:SLEW <number>

Parameter:	<decimal> rate in pressure units/second
Short form:	SOUR:SLEW <number>
Function:	Selects the pressure rate used when value rate is selected.
Default:	100

Query Syntax

:SOURCE[:PRESsure]:SLEW?

Short form:	:SOUR:SLEW?
Function:	Query rate value
Response:	Decimal number representing rate value in selected units/second

Description

When the controller rate is selected as value, this command is used to set the controllers rate in selected units/second.

Example

```
TX> :SOUR:PRES:SLEW?
RX> :SOUR:PRES:SLEW 2.0000000
```

Change current unit of mbar to bar and query again:

```
TX> :SOUR:PRES:SLEW?
RX> :SOUR:PRES:SLEW 0.0020000
```

To set slew rate to other values, such as:

```
TX> :SOUR:PRES:SLEW 4
TX> :SOUR:PRES:SLEW?
RX> :SOUR:PRES:SLEW 4.0000000
TX> :SOUR:PRES:SLEW max
TX> :SOUR:PRES:SLEW?
RX> :SOUR:PRES:SLEW 99999999.0000000
TX> :SOUR:PRES:SLEW min
TX> :SOUR:PRES:SLEW?
RX> :SOUR:PRES:SLEW 0.0
```

:SOUR:PRES:SLEW:MODE

Command Syntax

:SOURCE[:PRESsure]:SLEW:MODE <enumerated>

Parameter:	<enumerated>
	MAXimum - maximum rate
	LINear - user selected linear rate
Short form:	SOUR:SLEW:MODE <enumerated>
Function:	Select the rate the controller should use to achieve set-point.
Default:	MAXimum

Query Syntax

:SOURCE[:PRESsure]:SLEW:MODE?

Short form:	:SOUR:SLEW:MODE?
Function:	Query rate mode
Response:	MAX for maximum rate
	LIN for user defined rate

Description

The controller can operate in two rate modes - maximum and value. In maximum rate the controller tries to achieve set-point as quickly as possible. In value mode the controller achieves the set-point at a user selected rate.

<u>Example</u>

```
TX> :SOUR:PRES:SLEW:MODE?
RX> :SOUR:PRES:SLEW:MODE MAX
```

Can be set to user defined rate:

```
TX> :SOUR:PRES:SLEW:MODE linear
TX> :SOUR:PRES:SLEW:MODE?
RX> :SOUR:PRES:SLEW:MODE LIN
```

:SOUR:PRES:SLEW:OVER

Command Syntax

:SOURCE[:PRESsure]:SLEW:OVERshoot[:STATe] <Boolean>

Parameter:	<boolean>		
	0	-	overshoot not allowed
	1	-	overshoot allowed

Short form:	SOUR:SLEW:OVER <Boolean>
Function:	Selects pressure overshoot to `allowed’ or `not allowed’.
Default:	1 - overshoot allowed

Query Syntax

:SOURCE[:PRESsure]:SLEW:OVERshoot[:STATe]?

Short form:	:SOUR:SLEW:OVER?
Function:	Query overshoot state
Response:	0 - overshoot not allowed
	1 - overshoot allowed

Description

The controller can reach the set-point in one of two modes:

Overshoot `not allowed’, the controller changes the pressure to near the set-point. The rate of pressure change slows when approaching the set-point to avoid overshoot.

Overshoot `allowed’, the controller achieves set-point as fast as possible and, when approaching the set-point, may overshoot or undershoot.

Example

```
TX> :SOUR:PRES:SLEW:OVER?
RX> :SOUR:PRES:SLEW:OVER:STAT 0

TX> :SOUR:PRES:SLEW:OVER 1
TX> :SOUR:PRES:SLEW:OVER?
RX> :SOUR:PRES:SLEW:OVER:STAT 1
```

STATus

The STATus subsystem supports the OPERation status register as defined in SCPI protocol.

:STAT:OPER:COND

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:STATus:OPERation:CONDition?

Short form:	:STAT:OPER:COND?
Function:	Query condition register
Response:	Contents of condition register

Description
Returns the contents of the 16 bit condition register, see section on status reporting.

:STAT:OPER:ENAB

Command Syntax

:STATus:OPERation:ENABle <integer>

Parameter:	<integer>	16 bit value to set enable bits

Short form:	STAT:OPER:ENAB <integer>
Function:	Controls the status operation enable register.
Default:	0
	minimum 0
	maximum 32767

Query Syntax

:STATus:OPERation:ENABle?

Short form:	:STAT:OPER:ENAB?
Function:	Query enable register
Response:	16 bit value of enable register.

Description
Controls the bits that pass through the status reporting system, see status reporting section.

:STAT:OPER:EVEN

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:STATus:OPERation:[EVENt]?

Short form:	:STAT:OPER?
Function:	Query event register
Response:	16 bit value of event register.

Description
Reads contents of event register, see status reporting section.

:STAT:OPER:PRES:COND

Command Syntax

n/a

Parameter:	
Short form:	
Function:	

Query Syntax

:STATus:PRESsure:OPERation:CONDition?

Short form:	:STAT:OPER:PRES:COND?
Function:	Reads the contents of the pressure condition register.
Response:	Contents of pressure condition register.

Description

Returns the contents of the 16 bit pressure condition register, see section on status reporting.

:STAT:OPER:PRES:ENAB

Command Syntax

:STATus:OPERation:PRESsure:ENABle <integer>

Parameter:	<integer> 16 bit value to set pressure enable bits
Short form:	STAT:OPER:PRES:ENAB <integer>
Function:	Controls the pressure status operation enable register.
Default:	0
	minimum0
	maximum 32767

Query Syntax

:STATus:OPERation:PRESsure:ENABle?

Short form:	:STAT:OPER:PRES:ENAB?
Function:	Query enable register.
Response:	16 bit value of pressure enable register.

Description

Controls the bits that pass through the status reporting system, see status reporting section.

:STAT:OPER:PRES:EVEN

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:STATus:OPERation:PRESsure:[:EVENt]?

Short form:	:STAT:OPER:PRES?
Function:	Reads contents of pressure event register
Response:	16 bit value of pressure event register.

Description
Reads contents of pressure event register, see status reporting section.

Status System Examples

<u>Example *SRE and *STB</u>

```
             //Explanation:
*SRE 255     //To enable every bit of SRE register;
FRED         //Deliberately type a wrong command, such as;
*STB?        //Now queries Status Byte;
*STB 68      //Error Message set EAV and SERIAL POLL set MSS, 2^2+2^6 = 68;
*STB 0       //Previous read STB clears it.
```

Refer to Figure 3-1.

<u>Example of Pressure Event generated SRQ</u>

```
TX> *SRE 128                        //Enable SRE bit 7
TX> :STAT:OPER:ENAB 1024            //Enable Operation Register, bit 10
TX> :STAT:OPER:PRES:ENAB 32767      //Enable Pressure register all 16 bit
TX> :STAT:OPER:PRES:EVEN?           //Query Pressure Event
RX> :STAT:OPER:PRES:EVEN 0          //No any even yet
TX> :SENS:PRES?                     //What's the controller pressure now?
RX> :SENS:PRES 1099.9993896         //It is 1100 mbar
TX> :OUTP 1                         //Switch controller on
TX> :SOUR:PRES 2000                 //To generate an in-limit event
RX> :SRQ 192                        //Receive an SRQ automatically
TX> :STAT:OPER:PRES:EVEN?           //What happened to Pressure event?
RX> :STAT:OPER:PRES:EVEN 4          //In-limit event happened, see Table 3-2
TX> :STAT:OPER:PRES:EVEN?           //Ask again will clear the event register
RX> :STAT:OPER:PRES:EVEN 0          //It is cleared indeed
```

<u>Example of *IDN?</u>

```
TX> *IDN?
RX> *IDN GE Druck,Pace5000 User Interface,58784,01.05.04
```

SYSTem

The SYSTem subsystem consists of general purpose commands.

:SYST:ERR

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:SYStem:ERRor?

Query	:ERRor?
Short form:	:SYST:ERR?
Function:	Gets next error from the error queue

Response: The follow list of errors are available

-102, “Syntax error”
-104, “Data type error”
-108, “Parameter not allowed”
-109, “Missing parameter”
-110, “Command Header Error”
-111, “Header Separator Error”
-112, “Program mnemonic too long”
-113, “Undefined header”
-114, “Header suffix out of range”
-120, “Numeric data error”
-121, “Invalid character in number”
-123, “Exponent too large”
-124, “Too many digits”
-128, “Numeric data not allowed”
-130, “Suffix error”
-131, “Invalid suffix”
-134, “Suffix too long”
-138, “Suffix not allowed”

-140, “Character data error”
-141, “Invalid character data”
-144, “Character data too long”
-148, “Character data not allowed”
-150, “String data error”
-151, “Invalid string data”
-158, “String data not allowed”
-200, “Execution error”
-201, “Invalid while in local”
-202, “Settings lost due to rtl”
-220, “Parameter error”
-222, “Data out of range”
-223, “Too much data”
-224, “Illegal parameter value”
-310, “System error”
-350, “Queue overflow”
-400, “Query error”
201 , “Query only”
202 , “No query allowed”
203 , “Parameter(s) not expected”
207 , “Enumerated value not in union”
208 , “Illegal number of parameters”
210 , “Run out of memory handle”
211 , “Unit not matched”
212 , “Unit not required”

Description

This command queries the error queue which holds up to five errors. The instrument returns the message “No error” when no more errors are in the queue.

Example

Returns the system error
To test the system, send a wrong command, such as:
TX> :SENS:PRES qwer
Ask for error:
TX> :SYST:ERR?
RX> :SYST:ERR -200,"Execution error;Query or command violation"
This query empties the error stack, another query returns “No error”.
TX> :SYST:ERR?

:SYST:DATE

Command Syntax

:SYSTem:DATE

Parameter:	Integer in date format.
Short form:	:SYST:DATE
Function:	Sets the date.

Query Syntax

:SYSTem:DATE?

Short form:	:SYST:DATE?
Function:	Queries date
Response:	Returns date setting.

Description

Replies with the date settings in year, month, and day order.

<u>Example</u>

The system date can be set by:

```
TX> :SYST:DATE 9, 5, 2
TX> :SYST:DATE?
RX> :SYST:DATE 9, 5, 2

TX> :SYST:DATE?
RX> :SYST:DATE 9, 4, 1
```

It returns current date in year, month, and day order.

:SYST:SET

Command Syntax

:SYSTem:SET

Parameter:	Enumerated (measure or control) and a decimal number
Short form:	:SYST:SET
Function:	This command only effective at switch-on condition and can set the controller on with a set-point.

Query Syntax

:SYSTem:SET?

Short form:	:SYST:SET?
Function:	Queries current system setting.
Response:	Returns current settings in either measure or control with current set-point value.

Description

<u>Example</u>

```
TX> :SYST:SET?
RX> :SYST:SET MEAS, 0.0
```

At the current system set, when the instrument is switched on, the controller will be in measurement mode and the set-point will be zero.
This command can set the instrument to control mode, immediately after switch-on, with a set-point to 100 mbar, by sending:

```
TX> :SYST:SET CONT, 100.0
TX> :SYST:SET?
RX> :SYST:SET CONT, 100.0000000
```

Note:

The controller will not change mode without a switch on/off process. During normal operations use the following command example:

```
TX> :SOUR 2000
TX> :OUTP 1
```

This switches on the controller immediately with a set-point of 2000 mbar.

:SYST:TIME

Command Syntax

:SYSTem:TIME

Parameter:	Integer in time format.
Short form:	:SYST:TIME
Function:	Sets the time.

Query Syntax

:SYSTem:TIME?

Short form:	:SYST:TIME?
Function:	Returns current time in hour, minute and second.
Response:	Sends in order: hour, minute and second.

Description

This command sets the time. The command query returns the current time in hour, minute and second order.

Example

```
TX> :SYST:TIME?
RX> :SYST:TIME 10, 8, 44
```

It returns current time in hour, minute and second order.

Time can be reset by:

```
TX> :SYST:TIME 11, 11, 2
TX> :SYST:TIME?
RX> :SYST:TIME 11, 11, 4
```

:SYST:COMM:SER:CONT

Command Syntax

:SYSTem:COMMunicate:SERial:CONTrol

Parameter:	Integer
Short form:	SYST:COMM:SER:CONT
Function:	Sets the serial communication handshaking of: 0: NONE 1: XON/XOFF 2: RTS/CTS

Query Syntax

:SYSTem:COMMunicate:SERial:CONTrol?

Short form:	SYST:COMM:SER:CONT?
Function:	Queries serial communication handshaking.
Response:	Returns an integer of 0, 1, 2.

Description

Selects serial communications interface handshaking. The command query requests the current serial communication handshaking.

Example:

```
TX> :SYST:COMM:SER:CONT?
RX> :SYST:COMM:SER:CONT 0
```

The current hardware handshaking is NONE

It can be set to RTS/CTS by sending:

```
TX> :SYST:COMM:SER:CONT 2
```

The integer of 0, 1, 2 represents:
0: NONE
1: XON/XOFF
2: RTS/CTS

:SYST:COMM:SER:BAUD

Command Syntax

:SYSTem:COMMunicate:SERial:BAUD

Parameter:	Integer
Short form:	SYST:COMM:SER:BAUD
Function:	Sets the baud rate ***Note:*** *The parameter must be a valid Baud-rate number.*

Query Syntax

:SYSTem:COMMunicate:SERial:BAUD?

Short form:	:SYST:COMM:SER:BAUD?
Function:	Queries current baud setting.
Response:	Current baud setting.

Description

This command instructs the instrument to set the baud rate. The command query returns the current baud rate set in the instrument.

<u>Example</u>

```
TX> :SYST:COMM:SER:BAUD?
RX> :SYST:COMM:SER:BAUD 9600
```
The current baud-rate is 9600 bit/second.

It can be set to another baud rate (example: 19200) by sending:

```
TX> :SYST:COMM:SER:BAUD 19200
```

Notes

1: The parameter must be a valid baud rate number:

2400	9600	38400	115200
4800	19200	576000	

2: Changing to a new baud rate causes a loss of communications until resetting the local PC RS232 to the new baud rate.

:SYST:COMM:SER:TYPE:PARity

Command Syntax

:SYSTem:COMMunicate:SERial:TYPE:PARity

Parameter:	Enumerate
Short form:	SYST:COMM:SER:TYPE:PAR
Function:	Sets parity, odd, even or none.

Note:
This command breaks the communication between the PC and the PACE instrument. No further query can be made until after resetting the pc to the same setting. The instrument cannot be brought back to local mode.

Query Syntax

:SYSTem:COMMunicate:SERial:TYPE:PARity?

Short form:	:SYST:COMM:SER:TYPE:PAR?
Function:	Queries current parity setting.
Response:	Returns odd, even or none.

Description
This command instructs the instrument to set serial communication parity. The command query gets the current parity setting.

<u>Example</u>

```
TX> :SYST:COMM:SER:TYPE:PAR?
RX> :SYST:COMM:SER:TYPE:PAR NONE
```
The current setting ignores the parity check.

It can be set to ODD and EVEN by sending:
```
TX> :SYST:COMM:SER:TYPE:PAR ODD
```
or
```
TX> :SYST:COMM:SER:TYPE:PAR EVEN
```

:SYST:COMM:GPIB:SELF:ADDR

Command Syntax

:SYSTem:COMMunicate:GPIB:SELF:ADDRess

Parameter:	Integer
Short form:	SYST:COMM:GPIB:SELF:ADDR
Function:	Sets the instrument's GPIB address.

Query Syntax

:SYSTem:COMMunicate:GPIB:SELF:ADDRess?

Short form:	:SYST:COMM:GPIB:SELF:ADDR?
Function:	Queries GPIB address
Response:	

Description

This command instructs the instrument to set a GPIB address. The query command gets the current address.

<u>Example</u>

Set address to 16:

```
TX> :SYST:COMM:GPIB:SELF:ADDR 16
TX> :SYST:COMM:GPIB:SELF:ADDR?
RX> :SYST:COMM:GPIB:SELF:ADDR 16

:TX> :SYST:COMM:GPIB:SELF:ADDR?
RX> :SYST:COMM:GPIB:SELF:ADDR 1
```

Current GPIB address is 1

:SYST:AREA

Command Syntax

:SYSTem:AREA

Parameter:	String
Short form:	SYST:AREA
Function:	Sets a group of parameters for an area of the world.

Query Syntax

:SYSTem:AREA?

Short form:	:SYST:AREA?
Function:	Queries area setting.
Response:	Sends:

EUROPE, or EUR
USA
JAPAN or JAP
ASIA
ROW (rest of the world)

Description

This command instructs the instrument to enable a group of default settings for an area of the world.

```
:SYST:AREA
TX> :SYST:AREA?
RX> :SYST:AREA EUR
```

The current area of use is set to Europe.

Can be set to another area, such as Japan:

```
TX> :SYST:AREA jap
TX> :SYST:AREA?
RX> :SYST:AREA JAP
```

Set to the rest of the world:

```
TX> :SYST:AREA row
TX> :SYST:AREA?
RX> :SYST:AREA ROW
```

:SYST:PASS:CDIS

Command Syntax

:SYSTem:PASSword:CDISable

Parameter:	2317100
Short form:	SYST:PASS:CDIS
Function:	Disables the calibration with a password.

Query Syntax

n/a

Short form:
Function:
Response:

Description

This command disables calibration with a password.

:SYST:PASS:CEN

Command Syntax

:SYSTem:PASSword:CENable:

Parameter:	2317100
Short form:	:SYST:PASS:CEN
Function:	Enables calibration, default condition - disabled (calibration is not allowed).

Query Syntax

n/a

Short form:
Function:
Response:

Description

This command enables calibration.

:SYST:PASS:CEN:STAT

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:SYSTem:PASSword:CENable:STATe?

Short form:	SYST:PASS:CEN:STAT?		
Function:	Gets the status of the calibration password.		
Response:	0	-	disabled
	1	-	enabled

Description
This query asks if the calibration is enabled or disabled.

Example
It can be enabled by sending command and password

```
TX> :SYST:PASS:CEN 2317100
TX> :SYST:PASS:CEN:STAT?
RX> :SYST:PASS:CEN:STAT 1
```

When enabled and a 3-point calibration can be carried out.

It can be disabled again by issuing a command and a password, such as:

```
TX> :SYST:PASS:CDIS 2317100
TX> :SYST:PASS:CEN:STAT?
RX> :SYST:PASS:CEN:STAT 0
```

:SYST:VERS

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

:SYStem:VERSion?

Short form:	:SYST:VERS?
Function:	Returns the SCPI version.
Response:	version number

Description
Replies with the SCPI version number.

Example

```
TX> :SYST:VERS?
RX> :SYST:VERS 1995.0
```

UNIT

The UNIT sub-system configures the instrument's pressure measurement units.

:UNIT:PRES

Command Syntax

:UNIT[:PRESsure] <name>

Parameter: <name>

Valid units:

ATM	USER4	USER3	USER2	USER1
TORR	PSI	PA	MPA	MMHG
MMH2O	MHG	MH2O	MBAR	LB/FT2
KPA	KG/M2	KG/CM2	INHG	INH2O60
INH2O4	INH2O	HPA	FTH2O60	FTH2O4
FTH2O	CMHG	CMH2O	BAR	

Note: *Take care when entering the units, the letter 'O' can easily be mistaken for the number '0' or vice versa.*

Short form :UNIT <name>
Function: Selects pressure units

Query Syntax

:UNIT[:PRESsure]?

Short form: :UNIT?
Function: Query what pressure units are selected
Response: name as above

Description

This command selects the current pressure units; USER1 to USER4 are the user defined units.

<u>Example</u>

```
TX> :UNIT:PRES?
RX> :UNIT:PRES MBAR
```

The current unit is mbar

It can be set to another unit, such as bar by sending:

```
TX> :UNIT:PRES bar
TX> :UNIT:PRES?
RX> :UNIT:PRES BAR
```

The reply to this query confirms the change, the front panel display can also confirm the change.

:UNIT:PRES:DEF

Command Syntax

:UNIT[:PRESsure]:DEFine[x] <string>,<number>

Parameter:	<string>	ASCII representation of unit name.	
	<number>	Conversion factor from Pascals to required units	

Short form:	:UNIT:DEF [x] <string>,<number>		
Function:	There are four user defined units on the instrument. This command defines the name and conversion factor to use.		
	minimum	string	“”
		number	0.0
	maximum	string	8 characters
		number	1.0e10
Default:	string	N/A	
	number	1000.0	

Query Syntax

:UNIT[:PRESsure]:DEFine?

Short form:	:UNIT:DEF[x]?
Function:	Query conversion factor
Response:	A string representing name and number corresponding to the conversion factor.

1	USER1
2	USER2
3	USER3
4	USER4

Description

This command defines the name and conversion factor for the special units. This conversion factor is from Pascals to the required units. An index of 1 is for USER1 unit and an index of 2 is for USER2 unit.

<u>Example</u>

x = 1, 2, 3 and 4 representing the four user-defined units.

By default:

```
TX> :UNIT:PRES:DEF?
RX> :UNIT:PRES:DEF "UserUnit1", 1000.0000000
TX> :UNIT:PRES:DEF1?
RX> :UNIT:PRES:DEF "UserUnit1", 1000.0000000
TX> :UNIT:PRES:DEF2?
RX> :UNIT:PRES:DEF2 "UserUnit2", 1000.0000000
TX> :UNIT:PRES:DEF3?
RX> :UNIT:PRES:DEF3 "UserUnit3", 1000.0000000
TX> :UNIT:PRES:DEF4?
RX> :UNIT:PRES:DEF4 "UserUnit4", 1000.0000000
```

The first parameter is the unit name and the second the equivalent pressure in Pascals.

Example - Defining ‘MyUnit’

Setting USER4 as ‘MyUnit’:

```
TX> :UNIT:PRES:DEF4 "MyUnit", 2000.0
TX> :UNIT:PRES:DEF4?
RX> :UNIT:PRES:DEF4 "MyUnit", 2000.0000000
```

Select ‘MyUnit’ for use:

```
TX> :UNIT:PRES user4
TX> :UNIT:PRES?
RX> :UNIT:PRES USER4
```

The front panel display shows “MyUnit” and uses it for measurement, set-point, full-scales and source pressures.

Example

The following query:

```
TX> :SENS:PRES?
RX> :SENS:PRES 5.0000187
```

Returns a number in ‘MyUnit’

To confirm:

```
TX> :UNIT:PRES?
RX> :UNIT:PRES USER4
```

To find conversion factor for ‘MyUnit’

```
TX> :UNIT:PRES:DEF4?
RX> :UNIT:PRES:DEF4 "MyUnit", 2000.0000000
```

Calculate equivalent in Pascals: 5.0000187 multiplied by 2000.0000000.

4.2 Common Commands

The commands identified with * are SCPI common commands.

*CLS

Command Syntax

*CLS

Parameter:	none
Short form	*CLS
Function:	This command clears the all queues.

Query Syntax

n/a

Parameter:
Short form:
Function:

Description

Clears all event and condition register, see status reporting section.

*ESE

Command Syntax

*ESE <integer>

Parameter:	integer 8 bit value of enable mask
Short form:	*ESE <integer>
Function:	Sets the Standard Event Status enable register.
	minimum 0
	maximum 255
Default:	0

Query Syntax

*ESE?

Short form:	*ESE?
Function:	Query Standard Event Status Enable register.
Response:	8 bit integer of contents of Standard Event Status Enable register.

Description
See Standard Event Group, section 3-2 and Figure 3-1.

*ESR

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

*ESR?

Short form:	*ESR?
Function:	Queries the Standard Event Status Register
Response:	8 bit integer of contents of Standard Event Status register.

Description
See Standard Event Group, section 3-2 and Figure 3-1.

*IDN?

Command Syntax

n/a

Parameter:	
Short form:	
Function:	

Query Syntax

*IDN?

Short form:	*IDN?
Function:	Queries the identification of the instrument.
Response:	A comma separated list containing manufacture, model, serial number and software version.

Description

Return identification

e.g., *IDN Druck,PACE,1234,01.00.00

*SRE

Command Syntax

*SRE <integer>

Parameter:	integer	8 bit value of enable mask
Short form:	*SRE <integer>	
Function:	Sets the Service Request Enable register.	
	minimum	0
	maximum	255
Default:	0	

Query Syntax

*SRE?

Short form:	*SRE?
Function:	Query Service Request Enable register.
Response:	8 bit integer of contents of Service Request Enable register.

Description

See Status Byte Group, section 3-4 and Figure 3-1.

*STB?

Command Syntax

n/a

Parameter:

Short form:
Function:

Query Syntax

*STB?

Short form:	*STB?
Function:	Queries the Status Register
Response:	8 bit integer of contents of Status register.

Description

See status reporting section.

4.3 Instrument Control Commands

The following commands identified with : are SCPI instrument control commands.

:GTL

Command Syntax

:GTL

Parameter:	none
Short form	:GTL
Function:	Go to local

Query Syntax

n/a

Parameter:
Short form:
Function:

Description

Takes the instrument out of local lockout mode; the key-pad on the instrument becomes active.

:LOC

Command Syntax

:LOC

Parameter: none

Short form :LOC
Function: Local mode

Query Syntax

n/a

Parameter:
Short form:
Function:

Description

Puts the instrument into local mode. The instrument will no longer be in remote mode and can be operated from the front panel.

intentionally left blank

5 ERRORS

Negative error numbers are used for standard SCPI errors. Positive error numbers are device specific errors. Following the error number, a message describes the error. An error, when detected, is held in the error queue.
When SYST:ERR? is sent any error in the error queue sets the error bit in the event status register.

Error numbers
Each error detected causes an error number with an error message to be returned as follows:

Error code	Error message	Error code	Error message
-102	-102, “Syntax error”	-128	-128, “Numeric data not allowed”
-104	-104, “Data type error”	-130	-130, “Suffix error”
-108	-108, “Parameter not allowed”	-131	-131, “Invalid suffix”
-109	-109, “Missing parameter”	-134	-134, “Suffix too long”
-110	-110, “Command Header Error”	-138	-138, “Suffix not allowed”
-111	-111, “Header Separator Error”	-140	-140, “Character data error”
-112	-112, “Program mnemonic too long”	-141	-141, “Invalid character data”
-113	-113, “Undefined header”	-144	-144, “Character data too long”
-114	-114, “Header suffix out of range”	-148	-148, “Character data not allowed”
-120	-120, “Numeric data error”	-150	-150, “String data error”
-121	-121, “Invalid character in number”	-151	-151, “Invalid string data”
-123	-123, “Exponent too large”	-158	-158, “String data not allowed”
-124	-124, “Too many digits”		

Table 5-1 Errors -100 to -199

Error code	Error message	Error code	Error message
-200	-200, “Execution error”	-222	-222, “Data out of range”
-201	-201, "Invalid while in local"	-223	-223, “Too much data”
-202	-202, "Settings lost due to rtl"	-224	-224, “Illegal parameter value”
-220	-220, “Parameter error”		

Table 5-2 Errors -200 to -299

Error code	Error message
-310	-310, “System error”
-350	-350, “Queue overflow”
-400	-400, “Query error”

Table 5-3 Errors -300 to -400

Error code	Error message
201	201, “Query only”
202	202, “No query allowed”
203	203, “Paramerter(s) not expected”
207	207, "Emumerated value not in union"
208	208, “Illegal number of parameters”
210	210, “Run out of memory handle”
211	211, “Unit not matched”
212	212, “Unit not required”

Table 5-4 Errors +201 to +212