PROGRAMMING GUIDE

BÓSA

ARAGÓN PHOTONICS

Table of contents

0 Introduction

This programming guide gives information on how to control and program the BOSA equipment. The user is allowed to programmatically control the BOSA in three different ways:

- The Macro Editor Tool: it is a built-in functionality in the BOSA graphic user interface. It allows the user to program automatic measurements, set analysis functions and save results. Through simple scripts all BOSA functionalities are accessible. The Macro Editor Tool also enables communication with any external equipment via GPIB acting the BOSA as controller. The specific language syntax and SCPI commands are described in sections 3 and 4 in this manual.

- GPIB: the General Purpose Interface Bus is a common used way of controlling laboratory equipment. The GPIB interface of the BOSA allows remote control of the system allowing to automate measurements and to integrate the BOSA in your measurement set up. GPIB configuration and SCPI commands are described in sections 2 and 4 in this manual.

- Ethernet: the Ethernet interface is a standard interface used for personal computers to communicate over the network. The Ethernet connector on the rear panel of the BOSA allows remote communication through standard network cable. Network parameters and SCPI commands are described in sections 1 and 4 in this manual. Default password is 1234.

Macro Editor Tool, GPIB and Ethernet Interface use the same set of SCPI commands to operate the BOSA. A detailed description of the commands can be found in section 3.

1 Using the Ethernet Interface

When using the Ethernet interface the BOSA acts as a server. To operate, connect the Ethernet connector on the back of the BOSA to the LAN-network by using a standard network cable. The IP address of the BOSA should be configured to DHCP or manual (please refer to section 5.2 in the BOSA User Guide for details).

The port is 10000 on the TCP/IP protocol. Any command sent has to be followed by a read command. For non query commands the answer is “OK” or an error message. The error messages can be one of the followings: command error, parameter error or unit error.

The BOSA server will only admit one client at a time. Other clients trying to connect to the BOSA will not be taken into account.

2 Using the GPIB Interface

The GPIB (General Purpose Interface Bus) is an interface used for communication between personal computers and laboratory instruments, such as the BOSA. The IEEE-488 connector on the back of the BOSA allows for remote operation through a standard GPIB cable. The connector is a standard female 24-pin IEEE-488 connector to use with a standard shielded IEEE-488 cable.

Before you can operate the BOSA through GPIB interface you must assign the BOSA a device address that is unique from the other equipment attached to the bus.

Setting the Device Address

Under the Settings menu, select GPIB. The GPIB Configuration window appears. Select there a GPIB primary address for the BOSA different from the controller computer or the other instruments' addresses.

The default GPIB address of the BOSA is 4.

Returning to Local Control

When the BOSA is in remote control, the text of the Stop/Run button in the main screen of the BOSA will change to Local and all the main interface of the BOSA will be disabled. Press Local button if you wish to return the instrument to local control.

The complete set of commands that can be sent through GPIB interface are listed in section 3 of this manual. Additionally to that set of commands the BOSA will respond to the standard Identification query.

Identification query:

Syntax:	***IDN?**
Description:	Returns the system identification string of the BOSA.
Example of response:	ARAGON-PHOTONICS,BOSA-C,AC122201151010,V1.3.42 (Manufacturer: Aragon photonics, Model: Bosa-c, Hardware version: AC122201151010, Firmware version: V1.3.42)

Operation complete query:

Syntax:	***OPC?**
Description:	Returns an ASCII "1" when all pending overlapped operations have been completed.
Example of response:	1

3 Using the Macro Editor Tool

The *Macro Editor Tool* is a programming function that allows the user to program BOSA measurements conditions, set analysis functions and save results. It also enables communications with any external equipment acting the BOSA as controller. In the Macro editor are defined the following specific command categories:

- Variables and Arrays
- Conditional Statements and Functions
- BOSA Control Functions
- Mathematical Functions
- Date and Time Functions
- String Functions
- Explicit Conversion Functions

3.1 Macro Editor Command Summary

The following table gives an overview of the commands used in the Macro Editor, a description and a page reference for more detailed information about the particular command.

ARRAY FUNCTIONS		
Array **array**(int *length*)	Creates a new array with the specified *length*.	16
Void **sort**(string *arrayName*)	Sort the given array, identified by *arrayName*, into ascending order.	16
Void **reverse**(string *arrayName*)	Reverses the items of the given array, identified by *arrayName*,.	16
Int **length**(string *arrayName*)	Returns the length of the given array, identified by *arrayName*,.	16
Void **trim**(string *arrayName*)	Reduces the size of the given array, identified by *arrayName*, by eliminating the null values.	17

Void **plot** (string arrayName)	Plots the value of the arrayName in a chart	17
Void **plotXY**(string arrayNameX,string arrayNameY)	Plots the pair of values given by arrayNameX and arrayNameY in a XY chart.	17
CONDITIONAL STATEMENTS AND FUNCTIONS		

if(bool *condition*)
...
endif
	Performs a conditional statement.	17

double **iif**(bool *condition*, double *val1*, double *val2*)	Returns *val1* if *condition* is true, otherwise returns *val2*.	18
double **case**(bool *condition1*, double *val1*, bool *condition2*, double *val2*)	Returns the value after the first condition evaluated as true.	18

repeat
...
while(bool *condition*)
	If *condition* is true, the code execution jumps to the line after the repeat statement.	18

break	Jumps out of the actual repeat-while loop.	19
Void **wait**(int *value*)	Forces the BOSA to wait for the specified time before executing the next command. *Value* is given in milliseconds.	19
BOSA CONTROL INSTRUMENTS		
String **scpi**(string *command*)	Sends a scpi *command*. If it is a query command the answer is returned, otherwise it returns OK. An error string is returned if the command, parameters or units are not correct.	20
Void **sendgpib**(int *address*, string *command*)	Sends a GPIB *command* to an external instrument configured at the specified primary *address*. The command is specific for the external instrument.	20
String **querygpib**(int *address*, string *command*)	Sends a query GPIB *command* to an external instrument configured at the specified primary *address*. The command is specific for the external instrument.	20
Void **fwrite**(string *file*, string *value*)	Writes the *value* in the desired *file*.	21
Void **fwriteln**(string *file*, string *value*)	Writes a line with the *value* in the desired *file*.	21
Void **setvar**(string *name*, object *value*)	Creates a new variable identified by *name*, and initiates its *value*.	21
MATHEMATICAL FUNCTIONS		
Double **e**()	Returns the value of the e constant.	21

Double **min**(double *value1*, double *value2*)	Returns the smallest value of the two specified values.	26
Double **pow**(double *value1*, double *value2*)	Returns *value1* raised to *value2*.	26
Double **round**(double *value1*, int *value2*)	Returns *value1* rounded with the number of decimal digits specified by *value2*.	27
DATE & TIME FUNCTIONS		
DateTime **now**()	Returns the actual date time.	27
DateTime **today**()	Returns the actual date.	27
DateTime **mindate**()	Returns the minimum value for the date (00:00:00.0000000, 1st of January, 0001).	27
DateTime **maxdate**()	Returns the maximum value for the date (23:59:59.9999999, 31st of December, 9999).	28
String **monthname**()	Returns the name of the actual month.	28
DateTime **adddays**(dateTime *date*, int *days*)	Adds the specified amount of *days* to the specified *date*.	28
DateTime **addmonths**(dateTime *date*, int *months*)	Adds the specified amount of *months* to the specified *date.*	28
DateTime **addyears**(dateTime *date*, int *years*)	Adds the specified amount of *years* to the specified *date*.	28
DateTime **addhours**(dateTime *time*, int *hours*)	Adds the specified amount of *hours* to the specified *time*.	28
DateTime **addminutes**(dateTime *time*, int *minutes*)	Adds the specified amount of *minutes* to the specified *time*.	29
DateTime **addseconds**(dateTime *time*, int *seconds*)	Adds the specified amount of *seconds* to the specified *time*.	29
STRING FUNCTIONS		
String **concat**(string *value1*, string *value2*, string *value3*...)	Concatenates the strings *value1*, *value2*	30
String **formatnum**(double *value*, string *format*)	Returns a string with the *value* specified in the *format* specified.	30
String **formatdate**(dateTime *value*, string *format*)	Returns the date Time *value* specified in the *format* specified.	30
EXPLICIT CONVERSION FUNCTIONS		
Double **todouble**(object *value*)	Converts the specified *value* to a double.	31
Int **toint**(object *value*)	Converts the specified *value* to an integer.	31

Long **tolong**(object *value*)	Converts the specified *value* to a long.	31
Unsigned int **touint**(object *value*)	Converts the specified *value* to an unsigned integer.	31
Unsigned long **toulong**(object *value*)	Converts the specified *value* to an unsigned long.	31
DateTime **todatetime**(object *value*)	Converts the specified *value* to a date Time.	32
String **tostring**(object *value*)	Converts the specified *value* to a string.	32

3.2 Macro Editor Command Definitions

A new variable is declared just by assigning it a value. There is no need to define the type of the new variable and both functions and operators will perform automatic type conversion if needed.

Operator	Description	Example	Result
+	Addition	4+1	5
-	Subtraction	4-2	3
*	Multiplication	4*3	12
/	Division	4/3	1.3333
%	Modulo	4%3	1
&	Concatenation	1&2	12

Others:

Operator	Description	Example	Result
==	Comparison	4==4	True
!=	Negative comparison	4!=3	True
//	Comment		

Some examples:

- a=1
- b=2

- c = a + b (result: c = 3)
- c = a & b (result: c = 12)
- d = “3”
- f = a/d (result: f = 0.333)

3.2.1 Array Functions

3.2.1.1 Array

Syntax	**Array array(int *length*)**
Description	Creates a new array with the specified length.
Argument	‘length’ specifies the size of the array.
Example	myArray = array(10)
Comments	The first element is indexed with 0. When created, all elements are initialized to null values. To access an element: myArray[8]

3.2.1.2 Sort

Syntax	**Void sort(*arrayName*)**
Description	Sorts a given array into ascending order.
Argument	‘arrayName’ is the name of the array to be sorted.
Example	sort(myArray)
Comments	Only numeric arrays can be used.

3.2.1.3 Reverse

Syntax	**Void reverse(a*rrayName*)**
Description	Reverses the items of a given array.
Argument	‘arrayName’ is the name of the array to be reversed.
Example	reverse(myArray)
Comments	None.

3.2.1.4 Length

Syntax	**Int length(*arrayName*)**
Description	Returns the length of the given array.
Argument	‘arrayName’ is the name of the array.

Example	Length(myArray)
Comments	None.

3.2.1.5 Trim

Syntax	**Void trim(arrayName)**
Description	Reduces the size of the given array by eliminating the null values.
Argument	'arrayName' is the name of the array.
Example	trim(myArray)
Comments	Note that null value is not the same as zero value or empty (in strings).

3.2.1.6 Plot

Syntax	**Void plot(arrayName)**
Description	Plots the values of the array in a chart
Argument	'arrayName' is the name of the data array to display.
Example	plot(myArray)

3.2.1.7 PlotXY

Syntax	**Void plotXY(arrayNameX,arrayNameY)**
Description	Plots the pair of values of the two arrays in a XY chart
Argument	'arrayNameX' is the name of the data array to be displayed in the X axis and 'arrayNameY' is the data array to be displayed in the Y axis .
Example	plotXY(myArrayX,myArrayY)
Comments	ArrayX and arrayY lengths must be the same

3.2.2 Conditional Statements and functions

3.2.2.1 If

Syntax	**if(bool *condition*)** **...** **Endif**
Description	Performs a conditional statement.

Argument	'condition' can be a boolean value or expression.

Example	if(bool *condition*)
	  statement1
	  if(condition2)
	    statement3
	    ...
	  endif
	  statement2
	  ...
	Endif

Comments	Several condition statements can be nested.

3.2.2.2 Double iif

Syntax	**double iif(bool *condition*, double *val1*, double *val2*)**
Description	It returns the object val1 if condition is true, otherwise object val2.
Argument	'condition' can be a boolean value or expression. 'val1' is the result value if condition is true. 'val2' is the result value if condition is false.
Example	myMajor = iif(a>b,a,b)
Comments	It is expected that, at least one of the conditions is true.

3.2.2.3 Double case

Syntax	**double case(bool *condition1*, double *val1*, bool *condition2*, double *val2*)**
Description	Returns the parameter after the first condition parameter that evaluates to true.
Argument	'condition1' can be a boolean value or expression. 'val1' is the returned value if condition1 is true. 'condition2' can be a boolean value or expression. 'val2' is the returned value if condition2 is true.
Example	myCase = case(a<0,-1,a=0,0,a>0,1)
Comments	None.

3.2.2.4 Repeat-while

Syntax	**repeat**
	...
	while(bool *condition*)

Description	If condition is true, the code execution jumps to the line after the repeat statement.
Argument	‘condition’ can be a boolean value or expression.

Example	repeat
	        statement1
	        statement2
	        if(condition2)
	                repeat
	                        statement4
	                        …
	                while(condition3)
	break
	endif
	        statement3
	…
	while(bool *condition*)

Comments	Each statement is executed at least once.

3.2.2.5 Break

Syntax	**Break**
Description	Jumps out of the actual repeat-while loop
Argument	None.

Example	repeat
	        steatment 1
	        if(condition1)
	                break
	        endif
	while(condition2)

Comments	None.

3.2.2.6 Wait

Syntax	**Void wait(int *value*)**
Description	Forces the BOSA to wait for the specified time in milliseconds before executing the next command.
Argument	‘value’ specifies the number of milliseconds to wait.
Example	wait(1000)
Comments	None.

3.2.3 BOSA control instruments

3.2.3.1 String scpi(string command)

Syntax	**String scpi(string *command*)**
Description	Executes a scpi command. If it is a query command the answer is returned, otherwise it returns OK. An error string is returned if the command, parameters or units are not correct.
Argument	'command' is the SCPI command to manage the BOSA.
Example	myValue = scpi("sense:wavelength:center?")
Comments	Refer to section 4, *BOSA SCPI Commands*.

3.2.3.2 Void sendgpib(int address, string command)

Syntax	**Void sendgpib(int *address*, string *command*)**
Description	Sends a GPIB *command* to an external instrument configured at the specified primary *address*. The command is specific for the external instrument.
Argument	'address' specifies the GPIB primary address of the device to be controlled from the BOSA. 'command' is the command to be sent to the remote instrument.
Example	sendgpib(8,"state on")
Comments	None.

3.2.3.3 String querygpib(int address, string command)

Syntax	**String querygpib (int *address*, string *command*)**
Description	Sends a query GPIB *command* to an external instrument configured at the specified primary *address*. The command is specific for the external instrument.
Argument	'address' specifies the GPIB primary address of the device to be controlled from the BOSA. 'command' is the query command to be sent to the remote instrument.
Example	querygpib(8,"*IDN?")
Comments	None.

3.2.3.4 *Void fwrite(string file, string value)*

Syntax	**Void fwrite(string *file*, string *value*)**
Description	Writes the value in the desired file.
Argument	'file' specifies the complete path of the file. 'value' contents the string to be wrote in the file.
Example	Fwrite("D:\User Files\BOSA Data Files\myFile.txt","hello")
Comments	If the file does not exist, it will be created. If omitted, default path is 'D:\User Files\'. If the file already exists, lines will be added to the file.

3.2.3.5 *Void fwriteln(string file, string value)*

Syntax	**Void fwriteln(string *file*, string *value*)**
Description	Writes a line with the value in the desired file.
Argument	'file' specifies the complete path of the file. 'value' contents the line to be wrote in the file.
Example	fwriteln("D:\User Files\BOSA Data Files\myFile.txt","hello")
Comments	If the file does not exist, it will be created. If omitted, default path is 'D:\User Files\'. If the file already exists, lines will be added to the file.

3.2.3.6 *Void setvar(string name, object value)*

Syntax	**Void setvar(string *name*, object *value*)**
Description	Creates a new variable, and initiates its value.
Argument	'name 'specifies the name of the new variable. 'value' sets the value for the new variable.
Example	Setvar("myVar",155)
Comments	The new variable can be a double, an integer, a string, etc. Note the difference between: 'setvar("myVar",155)' and 'setvar("myVar","155")'. In the first case, myVar is an integer; in the second case myVar is a string.

3.2.4 Mathematical functions

3.2.4.1 *Double e()*

Syntax	**Double e()**
Description	Returns the value of the e constant.

Argument	None.
Example	myEE = e()*e()
Comments	None.

3.2.4.2 Double pi()

Syntax	**Double pi()**
Description	Returns the value of the Pi constant.
Argument	None.
Example	myArea = pi()*myRad*myRad
Comments	None.

3.2.4.3 Doble c()

Syntax	**Doble c()**
Description	Returns the speed of light in vacuum.
Argument	None.
Example	energy = mass*c()*c()
Comments	In IS units: metres/second, m/s.

3.2.4.4 Double sin(double value)

Syntax	**Double sin(double *value*)**
Description	Returns the sine of the specified angle.
Argument	‘value’ sets the angle, in radians.
Example	mySine = sin(myAngle)
Comments	None.

3.2.4.5 Double cos(double value)

Syntax	**Double cos(double *value*)**
Description	Returns the cosine of the specified angle.
Argument	‘value’ sets the angle, in radians.
Example	myCosine = cos(myAngle)
Comments	None.

3.2.4.6 Double tan(double value)

Syntax	**Double tan(double *value*)**
Description	Returns the tangent of the specified angle.

Argument	‘value’ sets the angle, in radians.
Example	myTangent = tan(myAngle)
Comments	None.

3.2.4.7 Double sec(double value)

Syntax	**Double sec(double *value*)**
Description	Returns the secant of the specified angle.
Argument	‘value’ sets the angle, in radians.
Example	mySecant = sec(myAngle)
Comments	None.

3.2.4.8 Double csc(double value)

Syntax	**Double csc(double *value*)**
Description	Returns the cosecant of the specified angle.
Argument	‘value’ sets the angle, in radians.
Example	myCosecant = csc(myAngle)
Comments	None.

3.2.4.9 Double cot(double value)

Syntax	**Double cot(double *value*)**
Description	Returns the cotangent of the specified angle.
Argument	‘value’ sets the angle, in radians.
Example	myCotangent = cot(myAngle)
Comments	None.

3.2.4.10 Double asin(double value)

Syntax	**Double asin(double *value*)**
Description	Returns the arcsine of the specified value, in radians.
Argument	‘value’ sets the sin value.
Example	myArcsine = asin(myAngle)
Comments	None.

3.2.4.11 Double acos(double value)

Syntax	**Double acos(double *value*)**
Description	Returns the arccosine of the specified value, in radians.

Argument	'value' sets the cos value.
Example	myArcosine = acos(myAngle)
Comments	None.

3.2.4.12 Double atan(double value)

Syntax	**Double atan(double *value*)**
Description	Returns the arctangent of the specified value, in radians.
Argument	'value' sets the tan value.
Example	myArctangent = atan(myAngle)
Comments	None.

3.2.4.13 Double sinh(double value)

Syntax	**Double sinh(double *value*)**
Description	Returns the hyperbolic sine of the specified angle.
Argument	'value' sets the angle, in radians.
Example	myHyperSine= sinh(myAngle)
Comments	None.

3.2.4.14 Double cosh(double value)

Syntax	**Double cosh(double *value*)**
Description	Returns the hyperbolic cosine of the specified angle.
Argument	'value' sets the angle, in radians.
Example	myHyperCosine= cosh(myAngle)
Comments	None.

3.2.4.15 Double tanh(double value)

Syntax	**Double tanh(double *value*)**
Description	Returns the hyperbolic tangent of the specified angle.
Argument	'value' sets the angle, in radians.
Example	myHyperTan = tanh(myAngle)
Comments	None.

3.2.4.16 Double abs(double value)

Syntax	**Double abs(double *value*)**
Description	Returns the absolute value of the specified value.

Argument	'value' sets the numeric value.
Example	myAbsolute = abs(myValue)
Comments	None.

3.2.4.17 Double sqrt(double value)

Syntax	**Double sqrt(double *value*)**
Description	Returns the square root of the specified value
Argument	'value' sets the numeric value.
Example	myRoot = sqrt(myValue)
Comments	Numeric value must be positive or 0.

3.2.4.18 Int ceil(double value)

Syntax	**Int ceil(double *value*)**
Description	Returns the smallest integer greater than the specified *value*.
Argument	'value' sets the numeric value.
Example	myInteger = ceil(myValue)
Comments	None.

3.2.4.19 Int floor(double value)

Syntax	**Int floor(double *value*)**
Description	Returns the largest integer smaller than the specified *value*.
Argument	'value' sets the numeric value.
Example	myInteger = floor(myValue)
Comments	None.

3.2.4.20 Double exp(double value)

Syntax	**Double exp(double *value*)**
Description	Returns e risen to a power indicated by *value*.
Argument	'value' sets the numeric value.
Example	myPower = exp(myValue)
Comments	None.

3.2.4.21 Double log10(double value)

Syntax	**Double log10(double *value*)**
Description	Returns the decimal logarithmic of the specified value.

Argument	'value' sets the numeric value for the logarithm.
Example	myDecLog = log10(myValue)
Comments	None.

3.2.4.22 Double log(double value1, [double value2])

Syntax	**Double log(double *value1*, [double *value2*])**
Description	Returns the logarithmic of the specified value, in the specified base.
Argument	'value1' sets the numeric value for the logarithm. 'value2' sets the base of the logarithm. If omitted, base is 'e'.
Example	myLog = log(myValue,myBase)
Comments	None.

3.2.4.23 Double max(double value1, double value2)

Syntax	**Double max(double *value1*, double *value2*)**
Description	Returns the largest value of the two specified values.
Argument	'value1' sets the first numeric value. 'value2' sets the second numeric value.
Example	myMaximum = max(a,b)
Comments	None.

3.2.4.24 Double min(double value1, double value2)

Syntax	**Double min(double *value1*, double *value2*)**
Description	Returns the smallest value of the two specified values.
Argument	'value1' sets the first numeric value. 'value2' sets the second numeric value.
Example	myMinimum = min(a,b)
Comments	None.

3.2.4.25 Double pow(double value1, double value2)

Syntax	**Double pow(double *value1*, double *value2*)**
Description	Returns the power of the value in the specified base.
Argument	'value1' sets the the base. 'value2' sets the power.
Example	myValue = pow(myBase,myPower)
Comments	None.

3.2.4.26 Double round(double value1, int value2)

Syntax	**Double round(double *value1*, int *value2*)**
Description	Returns a rounded value.
Argument	'value1' sets the numeric value. 'value2' sets the number of decimal digits.
Example	myRounded = round(myValue,2)
Comments	None.

3.2.5 Date and Time functions

3.2.5.1 DateTime now()

Syntax	**DateTime now()**
Description	Returns the actual date and time.
Argument	None.
Example	thisMoment = now()
Comments	The default format is described as follow: H: hour (24), h: hour (12), m: minutes, s: seconds, t: AM/PM d: day, M: month, y: year. 'MM/dd/yyyy hh:mm:ss tt'

3.2.5.2 DateTime today()

Syntax	**DateTime today(**
Description	Returns the actual date.
Argument	None.
Example	thisDay = today()
Comments	None.

3.2.5.3 DateTime mindate()

Syntax	**DateTime mindate()**
Description	Returns the minimum value for the date (00:00:00.0000000, 1st of January, 0001).
Argument	None.
Example	minMoment = mindate()
Comments	None.

3.2.5.4 DateTime maxdate()

Syntax	**DateTime maxdate()**
Description	Returns the maximum value for the date (23:59:59.9999999, 31st of December, 9999).
Argument	None.
Example	maxMoment = maxdate()
Comments	None.

3.2.5.5 String monthname()

Syntax	**String monthname()**
Description	Returns the name of the actual month.
Argument	None.
Example	thisMonth = monthname()
Comments	None.

3.2.5.6 DateTime adddays(dateTime date, int value)

Syntax	**DateTime adddays(dateTime *date*, int *value*)**
Description	Adds the specified amount of days to the specified date.
Argument	'date' represents the original date. 'value' sets the number of days to be added, can be negative

Example	myDate=today()
	newDate=adddays(myDate,15)

Comments	None.

3.2.5.7 DateTime addmonths(dateTime date, int value)

Syntax	**DateTime addmonths(dateTime *date*, int *value*)**
Description	Adds the specified amount of months to the specified date
Argument	'date' represents the original date. 'value' sets the number of months to be added, can be negative

Example	myDate=today()
	newDate=addmonths(myDate,6)

Comments	None.

3.2.5.8 DateTime addyears(dateTime date, int value)

Syntax	**DateTime addyears(dateTime *date*, int *value*)**
Description	Adds the specified amount of years to the specified date.

Argument	'date' represents the original date. 'value' sets the number of years to be added, it can be negative.
Example	addyears(myTime,1)
Comments	None.

3.2.5.9 DateTime addhours(dateTime time, int value)

Syntax	**DateTime addhours(dateTime *time*, int *value*)**
Description	Adds the specified amount of hours to the specified time
Argument	'time' represents the original time. 'value' sets the number of hours to be added, it can be negative.
Example	addhours(myTime,5)
Comments	None.

3.2.5.10 DateTime addminutes(dateTime time, int value)

Syntax	**DateTime addminutes(dateTime *time*, int *value*)**
Description	Adds the specified amount of minutes to the specified time
Argument	'time' represents the original time. 'value' sets the number of minutes to be added it can be negative.
Example	addminutess(myTime,20)
Comments	None.

3.2.5.11 DateTime Addseconds(dateTime time, int value)

Syntax	**DateTime Addseconds(dateTime *time*, int *value*)**
Description	Adds the specified amount of seconds to the specified time.
Argument	'time' represents the original time. 'value' sets the number of seconds to be added, it can be negative.
Example	addseconds(myTime,200)
Comments	None.

3.2.6 String functions

3.2.6.1 String concat(string value1, string value2, string value3…)

Syntax	**String concat(string *value1*, string *value2*, string *value3*...)**
Description	Concatenates the given strings
Argument	V*alue#* represents the strings to be added.
Example	myString = concat(strinng1,string2,string3)
Comments	The same effect is obtained by using the concatenate strings binary operator &. Example: myString = strinng1 & string2 & string3

3.2.6.2 String formatnum(double value, string format)

Syntax	**String formatnum(double *value*, string *format*)**
Description	Returns a string with the value specified in the format specified.
Argument	'value' represents the number to be formatted. 'format' sets the format of the number. For example "#.000".
Example	myString = formatnum(myValue, "#.000")
Comments	If value is 0.00123456 '#.0000': 0.0012 'e3': 1.235e-3

3.2.6.3 String formatdate(dateTime value, string format)

Syntax	**String formatdate(dateTime *value*, string *format*)**
Description	Returns the datetime specified in the format specified.
Argument	'value' represents the datetime to be formatted. 'format' sets the format of the date-time. For example "hh:mm:ss".
Example	myString = formatdate(myDateTime, "hh:mm:ss"
Comments	The format is described as follows: H: hour (24), h: hour (12), m: minutes, s: seconds, t: AM/PM d: day, M: month, y: year Example: 'hh:mm:ss tt dddd/dd/MM/yyyy'

3.2.7 Explicit conversion functions

3.2.7.1 Double todouble(object value)

Syntax	**Double todouble(object *value*)**
Description	Converts the specified value to a double.
Argument	‘value’ represents the object to be converted.
Example	myDouble = todouble(“0.12345”)
Comments	None.

3.2.7.2 Int toint(object value)

Syntax	**Int toint(object *value*)**
Description	Converts the specified value to an integer.
Argument	‘value’ represents the object to be converted.

Example	myInteger = toint(myDouble)
	myInteger = toint(“230”)

Comments	None.

3.2.7.3 Long tolong(object value)

Syntax	**Long tolong(object *value*)**
Description	Converts the specified value to a long integer,
Argument	‘value’ represents the object to be converted.

Example	myLong = tolong(myInteger)
	myLong = tolong(“230”)

Comments	None.

3.2.7.4 Unsigned int touint(object value)

Syntax	**Unsigned int touint(object *value*)**
Description	Converts the specified value to an unsigned integer.
Argument	‘value’ represents the object to be converted.
Example	myUInt = touint(myInteger)
Comments	None.

3.2.7.5 Unsigned long toulong(object value)

Syntax	**Unsigned long toulong(object *value*)**
Description	Converts the specified value to an unsigned long integer.

Argument	'value' represents the object to be converted.
Example	myULong = toulong(myInteger)
Comments	None.

3.2.7.6 DateTime todatetime(object value)

Syntax	**DateTime todatetime(object *value*)**
Description	Converts the specified value to a date Time
Argument	'value' represents the object to be converted.
Example	myDateTime = todatetime("10/10/2004 08:00:00 AM")
Comments	None.

3.2.7.7 String tostring(object value)

Syntax	**String tostring(object *value*)**
Description	Converts the specified value to a string.
Argument	'value' represents the object to be converted.
Example	myString = tostring(myInteger)
Comments	None.

4 BOSA SCPI Commands

The commands used by the BOSA are defined according to the Standard Commands for Programmable Instruments (SCPI) and are arranged in a command tree, also called subsystems. Every subsystem contains all commands belonging to a specific topic.

Some conventions related to the SCPI commands are:

- The part of the command shown in uppercase represents the short form of the command. The commands are case insensitive.
- Values to be input are indicated by angle brackets (<>) and are separated from the command by a colon as shown in the command syntax
- Optional values and portions of syntax are indicated by square brackets ([ ]).
- The bar (|) shows an either-or choice of data, for example a|b means either a or b, but not both simultaneously.

4.1 BOSA SCPI Command Summary

The following table gives an overview of the command tree. The commands and a page reference for more detailed information are also shown. Some commands are application specific and only work if the application is running. Their availability depends on your BOSA option (See section 2 in the User´s Guide). The following abbreviations are used to indicate the command applicability: BOSA (B), BOSA PHASE (BP), TUNABLE LASER (TLS), COMPONENT ANALYZER (CA).

INSTRUMENT SUBSYSTEM COMMANDS			
INSTrument:STATe:MODE ?	Queries the current application.	B,TLS,CA	42
INSTrument:STATe:MODE <MAIN|BOSA|TLS|CA>	Sets the application. To switch between applications you must always start from the MAIN	B,TLS,CA	42

	window.		
INSTrument:STATe:<HOLD?|RUN?>	Queries the current measurement state, and reads whether the BOSA is on.	B,CA	42
INSTrument:STATe:<HOLD|RUN> <1|0>	Sets the BOSA's state: HOLD or RUN.	B, CA	42
DISPLAY SUBSYSTEM COMMANDS			
DISPlay[:WINDow]:TRACe:Y[:SCALe]:AUTO [ONCE]	Sets an autoscale in Y axis.	B,CA	43
DISPlay[:WINDow]:TRACe:Y[:SCALe]:BOTTom <value> [DBM|MW]	Sets the lowest value of the vertical axis.	B,CA	43
DISPlay[:WINDow]:TRACe:Y[:SCALe]:BOTTom?	Queries the lowest value for of the vertical axis.	B,CA	43
DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision <numeric_value> [DBM|MW]	Sets power per division of the vertical axis.	B,CA	43
DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision?	Queries power per division for the vertical axis.	B,CA	43
DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel <numeric_value> [DBM|MW]	Sets the reference level value in the vertical axis.	B,CA	44
DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel?	Queries the reference level value in the vertical axis.	B,CA	44
DISPlay[:WINDow]:TRACe:Y[:SCALe]:NORMalize <0|1|OFF|ON>	Turns the normalization of the Y scale on or off	CA	44
DISPlay[:WINDow]:TRACe:Y[:SCALe]: NORMalize?	Queries the normalization of the Y scale.	CA	44
DISPlay[:WINDow]:TRACe:Y:SPACing <LOGarithmic|LINear>	Sets the vertical scale to logarithmic or lineal.	B	44
DISPlay[:WINDow]:TRACe:Y:SPACing?	Queries the current vertical scale.	B	45
DISPlay[:WINDow]:TRACe:X WAVelength|FREQuency	Changes the horizontal axis to the specified units.	B,CA	45
DISPlay[:WINDow]:TRACe:X?	Returns the current units of the horizontal axis.	B,CA	45
DISPlay[:WINDow]:TRACe:STATe <A|B|M1|M2|M3|M4>,<ON|OFF>	Change the active trace	B,CA	45

DISPlay[:WINDow]:TRACe:STATe?	Queries de active trace	B, CA	45
DISPlay[:WINDow]:GRAPHICSELection <C+L | O>	Changes the active graphic.	B	45
DISPlay[:WINDow]:GRAPHICSELection ?>	Queries the active graphic.	B	46
DISPlay[:WINDow]:GRAPHICVIEW <C+L | O | O+C+L>	Expands to full screen the desired graphic.	B	46
DISPlay[:WINDow]:GRAPHICVIEW ?>	Queries the displayed graphic.	B	46
SENSE SUBSYSTEM COMMANDS			
SENSe:WAVelength:CENTer <value> [NM|PM|GHZ|THZ]	Sets the center value of the horizontal axis.	B,CA	46
SENSe:WAVelength:CENTer?	Queries the center value of the horizontal axis.	B,CA	47
SENSe:WAVelenght:SINGLE <1|0|ON|OFF>	Set a single or continuous sweep	TLS	47
SENSe:WAVelenght:SINGLE?	Queries the type of sweeping	TLS	47
SENSe:WAVelenght:SMOOTH <numeric_value> [NM|PM|GHZ|THZ]	Performs a smooth operation over the active trace	CA	47
SENSe:WAVelenght:SMOOTH?	Queries the current smoothing value	CA	47
SENSe:WAVelength:SPAN <numeric_value> [NM|PM|GHZ|THZ]	Sets the span value of the horizontal axis.	B,CA	48
SENSe:WAVelength:SPAN?	Queries the span value of the horizontal axis.	B,CA	48
SENSe:WAVelenght:SPEED <numeric_value> [NM|PM|GHZ|THZ]	Sets the sweep speed for the internal tunable laser	TLS	48
SENSe:WAVelenght:SPEED?	Queries the current sweep speed of the internal tunable laser.	TLS	48
SENSe:WAVelength:STATic <numeric_value> [NM|PM|GHZ|THZ]	Sets the output wavelength for the internal tunable laser	TLS	48
SENSe:WAVelength:STATic?	Queries the current	TLS	48

	tuned wavelength for the internal tunable laser.		
SENSe:WAVelength:STARt <numeric_value> [NM|PM|GHZ|THZ]	Sets the minimum (left) value of the horizontal axis or the starting wavelength for a sweep.	B,CA,TLS	49
SENSe:WAVelength:STARt?	Queries the minimum (left) value of the horizontal axis or the starting wavelength for a sweep.	B,CA,TLS	49
SENSe:WAVelength:STOP <numeric_value> [NM|PM|GHZ|THZ]	Sets the maximum (right) value of the horizontal axis or the stopping wavelength for a sweep.	B,CA,TLS	49
SENSe:WAVelength:STOP?	Queries the maximum (right) value of the horizontal axis or the stopping wavelength for a sweep.	B,CA,TLS	49
SENSe:WAVelength:RESolution <numeric value>	Sets the BOSA resolution bandwidth	B	49
SENSe:WAVelength:RESolution?	Queries the BOSA sampling rate.	B	50
SENSe:WAVelength:SMODe <HR|HS>	Sets the measurement speed mode	CA	50
SENSe:WAVelength:SMODe?	Queries the measurement speed mode	CA	50
SENSe:AVERage:COUNt <4|8|12|32|CONT>	Sets the number of averages.	B,CA	50
SENSe:AVERage:COUNt?	Queries the current count of averages.	B,CA	50
SENSe:AVERage:STATe <0|1|OFF|ON>	Turns the averaging mode on or off.	B,CA	51
SENSe:AVERage:STATe?	Queries the state of averaging mode.	B,CA	51
SENSe:AVERage:CORRelate <0|1|ON|OFF>	Enables/disables the correlation function.	B,CA	51
SENSe:AVERage:CORRelate?	Queries the state of	B,CA	51

	the correlation function.		
SENSe:AVERage:CORRelate:CENTer <numeric_value> [NM]	Sets the center of the correlation zone on the active trace.	B,CA	51
SENSe:AVERage:CORRelate:CENTer?	Queries the center of the correlation zone on the active trace.	B,CA	52
SENSe:AVERage:CORRelate:SPAN <numeric_value> [NM]	Sets the width of the correlation zone on the active trace.	B,CA	52
SENSe:AVERage:CORRelate:SPAN?	Queries the width of the correlation zone on the active trace.	B,CA	52
SENSe:NOISezeroing	Performs a noise zeroing	B,CA	52
SENSe:SWITCH <0|1|ON|OFF>	Switches on or off the internal tunable laser	TLS	52
SENSe:SWITCH?	Queries if the internal tunable laser is switched on or off	TLS	53
SENSe:SWEEP <0|1|ON|OFF>	Starts or stops a sweep	TLS	53
SENSe:SWEEP?	Queries the sweeping status	TLS	53
SENSe:LASer <C+L | O>	Selects the internal laser to operate with through the TLS OUT front panel port.	TLS	53
SENSe:LASer?	Queries the actual internal laser that is being externally used for user purposes.	TLS	53
SENSe:BAND <C+L | O | O+C+L>	Change the measurement band.	B,CA	54
SENSe:BAND?>	Queries the actual measured band.	B,CA	54
INPUT SUBSYSTEM COMMANDS			
INPut:SPARameters <IL|RL|IL&RL>	Sets the type of measurement to display	CA	54
INPut:SPARameters?	Queries the type of measurement being displayed	CA	54
INPut:POLarization <1+2|1|2|1&2>	Sets the polarization	B	54

	option measurement		
INPut:POLarization <PDL|MAX|MIN|SIMUL>	Sets the displayed measure	CA (with optX30)	55
INPut:POLarization <INDEP|1|2|SIMUL>	Sets the displayed measure	CA (without optX30)	55
INPut:POLarization?	Queries the measured polarization	B,CA	55
INPut:POLarization:MUELLermode <ON|OFF|1|0>	Sets the Mueller Mode on or off	CA (with optX30)	55
INPut:POLarization:MUELLermode?	Queries if the Mueller Mode is active	CA (with optX30)	56
INPut:POWer?	Queries the total SUT power	B	56
CALCULATE SUBSYSTEM COMMANDS			
CALCulate:MARKer:AOFF	Disable the marker on the active trace.	B,CA	56
CALCulate:MARKer:STATe <ON|OFF|1|0>	Enables/disables marker on the active trace.	B,CA	56
CALCulate:MARKer:STATe?	Queries the state of the marker on the active trace.	B,CA	57
CALCulate:MARKer:MODe <TRCK|FIXX|FIXXY>	Sets the marker' behaviour.	B, CA	57
CALCulate:MARKer:MODe?	Queries he marker's behaviour.	B, CA	57
CALCulate:MARKer:MAXimum	Moves the marker on the active trace to the maximum power level of the trace.	B,CA	57
CALCulate:MARKer:MAXimum:NEXT	Moves the marker to the next power maximum.	B,CA	57
CALCulate:MARKer:MAXimum:RIGHT	Moves the marker to the next power maximum to the right.	B,CA	58
CALCulate:MARKer:MAXimum:LEFT	Moves the marker to the next power maximum to the left.	B,CA	58

CALCulate:MARKer:SCEN	Moves the marker on the active trace to the center of the screen.	B,CA	58
CALCulate:MARKer:X <numeric_value> [NM|PM|GHZ|THZ]	Moves the marker on the active trace to the specified point on the horizontal axis.	B,CA	58
CALCulate:MARKer:X?	Queries the horizontal coordinate of the marker on the active trace.	B,CA	58
CALCulate:MARKer:Y <numeric_value> [DBM|MW]	Moves the marker on the active trace to the specified point on the vertical axis.	B,CA	59
CALCulate:MARKer:Y?	Queries the vertical coordinate of the marker on the active trace.	59	59
CALCulate:MARKer:THREshold <numeric_value> [DB]	Sets the peak excursion value for the peak search functions.	59	59
CALCulate:MARKer:THREshold?	Queries the current peak excursion value for the peak search functions.	B,CA	59
CALCulate:MARKer:READout FREQuency | WAVelenght	Sets the horizontal units of the marker	B,CA	59
CALCulate:MARKer:READout?	Queries the horizontal units of the marker	B,CA	60
CALCulate:MARKer:SRLevel	Moves the reference level to the vertical value of the marker on the active trace.	B,CA	60
CALCulate:MARKer:POL?	In polarimetry, queries the S parameters (S1, S2, S3) of the marker	B	60
CALCulate:MARKer:FUNCtion:DELTa [:STATe] <ON|OFF|0|1>	Turns the delta marker function on or off.	B,CA	60

CALCulate:MARKer:FUNCtion:DELTa [:STATe]?	Queries the state of the delta marker function	B,CA	60
CALCulate:MARKer:FUNCtion:DELTa:RESet	Sets the marker offset to the current position of the marker	B,CA	60
CALCulate:MARKer:FUNCtion:DELTa:X:OFFSet?	Queries the horizontal offset between the marker and its reference	B,CA	61
CALCulate:MARKer:FUNCtion:DELTa:X:REFerece?	Queries the horizontal value of the reference marker	B,CA	61
CALCulate:MARKer:FUNCtion:DELTa:Y:OFFSet?	Queries the vertical offset between the marker and its reference	B,CA	61
CALCulate:MARKer:FUNCtion:DELTa:Y:REFerece?	Queries the horizontal value of the reference marker	B,CA	61
CALCulate:MARKer:FUNCtion:DELTa:POL?	In polarimetry, queries the S parameters (S1, S2, S3) of the offset	B	61
CALCulate:MARKer:FUNCtion:DELTa:ANG?	In polarimetry, queries the S angle between marker and offset	B	61
CALCulate:MAXimum[:STATe] <0|1|ON|OFF>	Enables/disables the Max Hold function.	B,CA	61
CALCulate:MAXimum[:STATe]?	Queries the state of the Max Hold function.	B,CA	62
CALCulate:MINimum[:STATe] ON|OFF	Enables/disables the Min Hold function.	B,CA	62
CALCulate:MINimum[:STATe]?	Queries the state of the Min Hold function.	B,CA	62
CALCulate:TPOWer ON|OFF	Enables/disables the power integral function	B,CA	63
CALCulate:TPOWer:[DATA]?	Returns the power integral.	B,CA	63

CALCulate:TPOWer:IRANge:UPPer<numeric_value> [NM|PM|GHZ|THZ]	Sets the upper bound for power integration.	B,CA	63
CALCulate:TPOWer:IRANge:UPPer?	Returns the upper bound for power integration.	B,CA	63
CALCulate:TPOWer:IRANge:LOWer <numeric_value> [NM|PM|GHZ|THZ]	Sets the lower bound for power integration.	B,CA	63
CALCulate:TPOWer:IRANge:LOWer?	Returns the lower bound for power integration.	B,CA	63
CALCulate:AUXINput:POWer?	Returns the power meter measurement from the Aux Input port.	TLS	64
TRACE SUBSYSTEM COMMANDS			
TRACe[:DATA]:COUNT?	Queries the number of points displayed on the screen.	B,CA	64
TRACe[:DATA]?	Queries the entire spectrum measured by the BOSA.	B,CA	64
TRACe[:DATA]:MAXimum:X?	Queries the X coordinate of the absolute maximum.	B,CA	65
TRACe[:DATA]:MAXimum:Y?	Queries the Y coordinate of the absolute maximum.	B,CA	65
FORMat[:DATA] <ASCII|REAL>,<length>	Sets the output data format for queries either to ASCII with a given length or REAL (64 bits).	B,CA	65
FORMat[:DATA]?	Returns the current output data format of the GPIB queries.	B,CA	65
MMEMORY SUBSYSTEM COMMANDS			
MMEMory:STORe:TRACe <name>.<type>,[ink_saving_mode]	Stores the traces displayed in a file in the specified format.	B,CA	66
MMEMory:DELete:TRACe <name>	Deletes the specified file	B,CA	66
MMEMory:LOAD:TRACe <M1|M2|M3|M4>,<name>	Loads the specified file	B,CA	66

4.2 BOSA SCPI Command Definitions

4.2.1 INSTrument Subsystem Commands

4.2.1.1 INSTrument:STATe:MODE ?

Syntax	**INSTrument:STATe:MODE ?**
Description	Queries the current application.
Response	'MAIN' if the unit is running the main menu, and <BOSA|TLS|CA> if it is running one of the applications.
Example	INST:STAT:MODE ?

4.2.1.2 INSTrument:STATe:MODE <MAIN/BOSA/TLS/CA>

Syntax	**INSTrument:STATe:MODE <MAIN|BOSA|TLS|CA>**
Description	Sets the application. To switch between applications you must always start from the MAIN window.
Example	INST:STAT:MODE BOSA

4.2.1.3 INSTrument:STATe:<HOLD?/RUN?>

Syntax	**INSTrument:STATe:<HOLD?|RUN?>**
Description	Queries the current measurement state, and reads whether the BOSA is on.
Response	'ON' if it is running, and 'OFF' if it is stopped.
Example	INST:STAT: RUN?

4.2.1.4 INSTrument:STATe:<HOLD/RUN>

Syntax	**INSTrument:STATe:<HOLD|RUN>** <1|0>
Description	Sets the BOSA's state: HOLD or RUN.
Argument	'HOLD' to freeze the trace on the screen, the BOSA continues working on a second plane and 'RUN' to start and stop the application.
Example	INST:STAT:RUN 1

4.2.2 DISPlay Subsystem Commands

4.2.2.1 DISPlay[:WINDow]:TRACe:Y[:SCALe]:AUTO [ONCE]

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:AUTO [ONCE]**
Description	Performs an auto scale in the vertical axis.
Example	DISP:TRAC:Y:AUTO

4.2.2.2 DISPlay[:WINDow]:TRACe:Y[:SCALe]:BOTTom

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:BOTTom <value> [DBM|MW]**
Description	Sets the lowest value of the vertical axis.
Argument	'value' is the lowest value of the vertical axis. It can be a numeric value between 20 and -80 dBm or the Ref Level, using the key 'rlev'. Units can be 'DBM' for logarithmic scale or 'MW' for linear scale.
Example	DISP:TRAC:Y:BOTT -70 DBM

4.2.2.3 DISPlay[:WINDow]:TRACe:Y[:SCALe]:BOTTom?

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:BOTTom?**
Description	Queries the lowest value of the vertical axis.
Response	The lowest value of the vertical axis, expressed in the current units.
Example	DISP:TRAC:Y:BOTT?

4.2.2.4 DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision <numeric_value> [DBM|MW]**
Description	Sets power per division of the vertical axis.
Argument	'numeric_value' is the power per division (max. 10 dB) for the vertical axis. Units are optional. Values must match the Y scale units.
Example	DISP:TRAC:Y:PDIV 10 DBM

4.2.2.5 DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision?

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision?**
Description	Queries power per division for the vertical axis.

Response	The power per division for the vertical axis, expressed in the current units.
Example	DISP:TRAC:Y:PDIV?

4.2.2.6 DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel <numeric_value> [DBM|MW]**
Description	Sets the reference level value in the vertical axis.
Argument	'numeric_value' is the value for the reference level.
Example	DISP:TRAC:Y:RLEV 0 DBM

4.2.2.7 DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel?

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel?**
Description	Queries the reference level value in the vertical axis.
Response	The vertical value for the reference level, expressed in the current units.
Example	DISP:TRAC:Y:RLEV?

4.2.2.8 DISPlay[:WINDow]:TRACe:Y[:SCALe]:NORMalize

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:NORMalize <0|1|OFF|ON>**
Description	Turns the normalization of the Y scale on or off.
Example	DISP:TRAC:Y:NORM 0

4.2.2.9 DISPlay[:WINDow]:TRACe:Y[:SCALe]:NORMalize?

Syntax	**DISPlay[:WINDow]:TRACe:Y[:SCALe]:NORMalize?**
Description	Queries the normalization of the Y scale.
Response	It returns 1 or On if the normalization is applied to the Y scale.
Example	DISP:TRAC:Y:NORM?

4.2.2.10 DISPlay[:WINDow]:TRACe:Y:SPACing

Syntax	**DISPlay[:WINDow]:TRACe:Y:SPACing <LOGarithmic|LINear>**
Description	Sets the vertical scale.
Argument	It can be 'logarithmic' or 'linear' for the vertical axis.
Example	DISP:TRAC:Y:SPAC LIN

4.2.2.11 DISPlay[:WINDow]:TRACe:Y:SPACing?

Syntax	**DISPlay[:WINDow]:TRACe:Y:SPACing?**
Description	Queries the vertical scale.
Response	It returns the current spacing of the Y axis: 'Spacing Logarithmic' or 'Spacing Linear'.
Example	DISP:TRAC:Y:SPAC?

4.2.2.12 DISPlay[:WINDow]:TRACe:X WAVelength|FREQuency

Syntax	**DISPlay[:WINDow]:TRACe:X WAVelength|FREQuency**
Description	Changes the x axis to the specified units, nm or GHz
Example	DISP:TRAC:X WAV

4.2.2.13 DISPlay[:WINDow]:TRACe:X?

Syntax	**DISPlay[:WINDow]:TRACe:X?**
Description	Returns the current units of the x axis.
Example	DISP:TRAC:X?

4.2.2.14 DISPlay[:WINDow]:TRACe:STATe

Syntax	**DISPlay[:WINDow]:TRACe:STATe <M1|M2|M3|M4>,<ON|OFF>**
Description	Changes the active trace.
Example	DISP:TRAC:STAT M2,ON
Comments	If trace M2 is activated and it does not exist, the Active trace is copied in trace M2. If the trace exist it becomes the active trace. The OFF state clear the selected trace

4.2.2.15 DISPlay[:WINDow]:TRACe:STATe?

Syntax	**DISPlay[:WINDow]:TRACe:STATe?**
Description	Queries the active trace.
Example	DISP:TRAC:STAT?

4.2.2.16 DISPlay[:WINDow]:GRAPHICSELection <C+L/O>

Syntax	**DISPlay[:WINDow]:GRAPHICSELection <C+L|O>**

Description	Changes the active graphic to C+L or O band graphic. The sent commands afterwards are enabled over the active graphic. (Only available for BOSA O+C+L)
Example	DISP:GRAPHSEL C+L

4.2.2.17 DISPlay[:WINDow]:GRAPHICSELection ?

Syntax	**DISPlay[:WINDow]:GRAPHICSELection <C+L|O>**
Description	Queries the active graphic. (Option only valid for BOSA O+C+L).
Response	'C+L' if the selected graph corresponds to the C+L band, 'O' if the selected graph corresponds to the O band.
Example	DISP:GRAPHSEL ?

4.2.2.18 DISPlay[:WINDow]:GRAPHICVIEW<C+L/O/O+C+L>

Syntax	**DISPlay[:WINDow]:GRAPHICVIEW <C+L|O|O+C+L>**
Description	Expands to full screen the desired graphic, when O+C+L option is sent, screen is divided into 2 graphs. O+C+L option is only available when simultaneous bands are measured, see command 4.2.3.38. (Only available for BOSA O+C+L)
Example	DISP:GRAPHVIEW C+L

4.2.2.19 DISPlay[:WINDow]:GRAPHICVIEW ?

Syntax	**DISPlay[:WINDow]:GRAPHICVIEW ?**
Description	Queries the displayed graphic. (Option only valid for BOSA O+C+L)
Response	'C+L' if the displayed graph corresponds to the C+L band, 'O' if the displayed graph corresponds to the O band, 'O+C+L' if simultaneous mode.
Example	DISP:GRAPHVIEW ?

4.2.3 SENSe Subsystem Commands

4.2.3.1 SENSe:WAVelength:CENTer

Syntax	**SENSe:WAVelength:CENTer <value> [NM|PM|GHZ|THZ]**
Description	Sets the center value of the horizontal axis.
Argument	It can be a numeric value or 'MAX'. If ´MAX´ is used, the

	maximum power level displayed in the screen is moved to the center.
Example	SENS:WAV:CENT 1550 NM

4.2.3.2 SENSe:WAVelength:CENTer?

Syntax	**SENSe:WAVelength:CENTer?**
Description	Queries the center value of the horizontal axis.
Response	The center value of the horizontal axis, expressed in nm or GHz
Example	SENS:WAV:CENT?

4.2.3.3 SENSe:WAVelength:SINGLE

Syntax	**SENSe:WAVelength:SINGLE 1|0|ON|OFF**
Description	Sets the type of sweeping for the internal tunable laser
Argument	'1' or ON set a single sweep, '0' or OFF set a continuous sweep.
Example	SENS:WAV:SINGLE ON

4.2.3.4 SENSe:WAVelength:SINGLE?

Syntax	**SENSe:WAVelength:SINGLE?**
Description	Queries the current type of sweeping of the internal tunable laser
Response	ON means a single sweep is set, OFF means a continuous sweep is set.
Example	SENS:WAV:SINGLE?

4.2.3.5 SENSe:WAVelength:SMOOTH

Syntax	**SENSe:WAVelength:SMOOTH <value> [NM|PM|GHZ|THZ]**
Description	Performs a smoothing operation over the active trace
Argument	The spectral width averaged in the smoothing
Example	SENS:WAV:SMOOTH 1 GHZ

4.2.3.6 SENSe:WAVelength:SMOOTH?

Syntax	**SENSe:WAVelength:SMOOTH?**
Description	Queries the current value of the smoothing operation
Response	The spectral width averaged in the smoothing
Example	SENS:WAV:SMOOTH?

4.2.3.7 SENSe:WAVelength:SPAN

Syntax	**SENSe:WAVelength:SPAN <numeric_value> [NM|PM|GHZ|THZ]**
Description	Sets the span value of the horizontal axis.
Argument	The span value of the horizontal axis.
Example	SENS:WAV:SPAN 2 NM

4.2.3.8 SENSe:WAVelength:SPAN?

Syntax	**SENSe:WAVelength:SPAN?**
Description	Queries the span value of the horizontal axis.
Response	The span value of the horizontal axis, expressed in nm or GHz.
Example	SENS:WAV:SPAN?

4.2.3.9 SENSe:WAVelength:SPEED

Syntax	**SENSe:WAVelength:SPEED <numeric_value> [NM|PM|GHZ|THZ]**
Description	Sets the sweep speed of the internal tunable laser
Argument	The sweep speed. Units are given per second.
Example	SENS:WAV:SPEED 2 NM

4.2.3.10 SENSe:WAVelength:SPEED?

Syntax	**SENSe:WAVelength:SPEED?**
Description	Queries the current sweep speed of the internal tunable laser.
Argument	The sweep speed. Units are given per second.
Example	SENS:WAV:SPEED?

4.2.3.11 SENSe:WAVelength:STAT

Syntax	**SENSe:WAVelength:STAT <numeric_value> [NM|PM|GHZ|THZ]**
Description	Sets the tuning wavelength for the internal tunable laser.
Argument	The tunable laser output wavelength
Example	SENS:WAV:STAT 1549 NM

4.2.3.12 SENSe:WAVelength:STAT?

Syntax	**SENSe:WAVelength:STAT?**
Description	Queries the tuned wavelength for the internal tunable laser.
Response	The tuned wavelength

Example	SENS:WAV:STAT?

4.2.3.13 SENSe:WAVelength:STARt

Syntax	**SENSe:WAVelength:STARt <numeric_value> [NM|PM|GHZ|THZ]**
Description	Sets the minimum (left) value of the horizontal axis or the starting wavelength for a sweep.
Argument	The value for the minimum in the horizontal axis or the starting wavelength for a sweep.
Example	SENS:WAV:START 1549 NM

4.2.3.14 SENSe:WAVelength:STARt?

Syntax	**SENSe:WAVelength:STARt?**
Description	Queries the minimum (left) value of the horizontal axis or the starting wavelength for a sweep.
Response	The value for the minimum in the horizontal axis or the starting wavelength for a sweep, expressed in nanaometers.
Example	SENS:WAV:START?

4.2.3.15 SENSe:WAVelength:STOP

Syntax	**SENSe:WAVelength:STOP <numeric_value> [NM|PM|GHZ|THZ]**
Description	Sets the maximum (right) value of the horizontal axis or the stopping wavelength for a sweep.
Argument	The value for the maximum in the horizontal axis.
Example	SENS:WAV:STOP? 1551 NM

4.2.3.16 SENSe:WAVelength:STOP?

Syntax	**SENSe:WAVelength:STOP?**
Description	Queries the maximum (right) value of the horizontal axis or the stopping wavelength for a sweep.
Response	The value for the maximum of the horizontal axis or the stopping wavelength for a sweep, expressed in nm or GHz.
Example	SENS:WAV:STOP?

4.2.3.17 SENSe:WAVelength:RESolution

Syntax	**SENSe:WAVelength:RESolution <value> [NM|PM|GHZ|THZ]**

Description	Sets the resolution bandwidth.
Argument	The spectral filter width.
Example	SENS:WAV:RES 1 GHZ

4.2.3.18 SENSe:WAVelength:RESolution?

Syntax	**SENSe:WAVelength:RESolution?**
Description	Queries the resolution bandwidth.
Response	The spectral filter width.
Example	SENS:WAV:RES?

4.2.3.19 SENSe:WAVelength:SMODe

Syntax	**SENSe:WAVelength:SMODe <HR|HS>**
Description	Change the speed mode of the component analyser measurement.
Argument	The value ‘HR’ stands for the High Resolution mode while the ‘HS’ sets the mode to the High Speed mode. Refer to the User Guide for further explaining.
Example	SENS:WAV:SMOD HR

4.2.3.20 SENSe:WAVelength:SMODe?

Syntax	**SENSe:WAVelength:SMODe?**
Description	Queries the actual measurement speed mode
Response	‘HR’ stands for the High Resolution mode while the ‘HS’ stands for the High Speed mode
Example	SENS:WAV:SMOD?

4.2.3.21 SENSe:AVERage:COUNt

Syntax	**SENSe:AVERage:COUNt 4|8|12|32|CONT**
Description	Sets the number of averages.
Argument	Available number of averages: 4, 8, 12, 32 or CONTINUOUS.
Example	SENS:AVER:COUN CONT

4.2.3.22 SENSe:AVERage:COUNt?

Syntax	**SENSe:AVERage:COUNt?**
Description	Queries the current number of averages.

Response	The current count of averages. It is recommended to use *wait* command to let the BOSA reach the number of averages set.
Example	SENS:AVER:COUN?

4.2.3.23 SENSe:AVERage:STATe

Syntax	**SENSe:AVERage:STATe 0|1|OFF|ON**
Description	Enables or disables the averaging mode.
Argument	'1' or 'ON' to start a new average, and '0' or 'OFF' to stop it.
Example	SENS:AVER:STAT ON

4.2.3.24 SENSe:AVERage:STATe?

Syntax	**SENSe:AVERage:STATe?**
Description	Queries the state of averaging mode.
Response	The current state of the averaging mode: 'ON' if averaging, 'OFF' if stopped.
Example	SENS:AVER:STAT?

4.2.3.25 SENSe:AVERage:CORRelate

Syntax	**SENSe:AVERage:CORRelate 0|1|ON|OFF**
Description	Enables or disables the correlation function in order to lock the trace relatively to a peak.
Argument	'1' or 'on' to enable the correlation; '0' or 'off' to disable it.
Example	SENS:AVER:CORR ON

4.2.3.26 SENSe:AVERage:CORRelate?

Syntax	**SENSe:AVERage:CORRelate?**
Description	Queries the state of the correlation function.
Response	'ON' if it is enabled, 'OFF' if not.
Example	SENS:AVER:CORR?

4.2.3.27 SENSe:AVERage:CORRelate:CENTer

Syntax	**SENSe:AVERage:CORRelate:CENTer <numeric_value> [NM]**
Description	Sets the center of the correlation zone.
Argument	<numeric_value> is the value for the center of the correlation zone.

Example	SENS:AVER:CORR:CENT 1550 NM
Comments	Referred to the active trace.

4.2.3.28 SENSe:AVERage:CORRelate:CENTer?

Syntax	**SENSe:AVERage:CORRelate:CENTer?**
Description	Queries the center of the correlation zone.
Response	The center of the correlation zone, expressed in nm or GHz.
Example	SENS:AVER:CORR:CENT?
Comments	Referred to the active trace.

4.2.3.29 SENSe:AVERage:CORRelate:SPAN

Syntax	**SENSe:AVERage:CORRelate:SPAN <numeric_value> [NM]**
Description	Sets the width of the correlation zone.
Argument	<numeric_value> is the value for the span of the correlation zone.
Example	SENS:AVER:CORR:SPAN 0.01 NM
Comments	Referred to the active trace.

4.2.3.30 SENSe:AVERage:CORRelate:SPAN?

Syntax	**SENSe:AVERage:CORRelate:SPAN?**
Description	Queries the span of the correlation zone.
Response	The span of the correlation zone, expressed in nm or GHz.
Example	SENS:AVER:CORR:SPAN?
Comments	Referred to the active trace.

4.2.3.31 SENSe:NOISezeroing

Syntax	**SENSe:NOISezeroing**
Description	Performs a noise zeroing.
Example	SENS:NOIS
Comments	This function carries out an immediate improvement in the noise pattern of the current spectrum.

4.2.3.32 SENSe:SWITCH

Syntax	**SENSe:SWITCH 1|0|ON|OFF**

Description	Switches on or off the internal tunable laser
Arguments	'1' or ON switch on the laser, '0' or OFF switch off the laser.
Example	SENS:SWITCH ON

4.2.3.33 SENSe:SWITCH?

Syntax	**SENSe:SWITCH?**
Description	Queries the internal tunable laser is switched on or off
Response	ON if the laser switched on, OFF if the laser is switched off.
Example	SENS:SWITCH?

4.2.3.34 SENSe:SWEEP

Syntax	**SENSe:SWEEP 1|0|ON|OFF**
Description	Start or stops a sweep
Arguments	'1' or ON starts the sweep,'0' or OFF stops the sweep.
Example	SENS:SWEEP ON

4.2.3.35 SENSe:SWEEP?

Syntax	**SENSe:SWEEP?**
Description	Queries the sweeping status of the laser
Response	ON if the laser is sweeping, OFF if the laser is stopped
Example	SENS:SWEEP?

4.2.3.36 SENSe:LASer <C+L / O>

Syntax	**SENSe:LASer <C+L | O>**
Description	Selects the internal tuneable laser to operate with through the TLS OUT port. User can choose C+L or O band laser source. (Only available for BOSA O+C+L)
Example	SENS:LAS C+L

4.2.3.37 SENSe:LASer ?

Syntax	**SENSe:LASer** ?
Description	Queries the actual internal laser that is being externally used for user purposes. (Only available for BOSA O+C+L)

Comments	This command is only available for the Tuneable Laser application.
Response	‘Active internal tunable laser: C+L’ or ‘O’
Example	SENS:LAS ?

4.2.3.38 SENSe:BAND <C+L / O / O+C+L>

Syntax	SENSe:BAND <C+L|O|O+C+L>
Description	Changes the measurement band to C+L, O or O+C+L band. BOSA application must be off to in order to change this option. Component Analyzer application does not support O+C+L option. (Only available for BOSA O+C+L)
Example	SENS:BAND O+C+L

4.2.3.39 SENSe:BAND ?

Syntax	SENSe:BAND ?
Description	Queries the actual measured band. (Only available for BOSA O+C+L)
Response	‘Active Measured Band: C+L’ or ‘O’
Example	SENS:BAND ?

4.2.4 INPut Subsystem Commands

4.2.4.1 INPut:SPARameters

Syntax	INPut:SPARameters [IL|RL|IL&RL]
Description	Selects the measurement to display. The options are Insertion Loss (IL) and Return Loss (RL).
Comments	This command is only available for the Component Analyzer application
Example	INP:SPAR IL&RL

4.2.4.2 INPut:SPARameters?

Syntax	INPut:SPARameters?
Description	Queries the type of measurement being displayed
Response	Possible responses are RL (return loss), IL (insertion loss) or IL&RL (insertion loss and return loss)

Comments	This command is only available for the Component Analyzer application
Example	INP:SPAR?

4.2.4.3 INPut:POLarization

Syntax	**INPut:POLarization 1+2|1|2|1&2**
Description	Selects the polarization measurement to be displayed.
Argument	It is possible to choose between the two orthogonal polarization states and display them separately or simultaneously.
Example	INP:POL 2

4.2.4.4 INPut:POLarization

Syntax	**INPut:POLarization PDL|MAX|MIN|SIMUL**
Description	Selects the polarization measurement to be displayed.
Comments	This command is only available for the Component Analyzer application with 230 option
Example	INP:POL PDL

4.2.4.5 INPut:POLarization

Syntax	**INPut:POLarization INDEP|1|2|SIMUL**
Description	Selects the polarization measurement to be displayed.
Comments	This command is only available for the Component Analyzer application without 230 option
Example	INP:POL SIMUL

4.2.4.6 INPut:POLarization?

Syntax	**INPut:POLarization?**
Description	Queries the current polarization configuration that is being displayed
Response	Depends on the current application and the hardware option installed.
Example	INP:POL?

4.2.4.7 INPut:POLarization:MUELLermode

Syntax	**INPut:POLarization:MUELLermode <ON|OFF|1|0>**

Description	Sets the advance polarization analysis on or off. Refer to the User Guide for further explaining.
Comments	This command is only available for the Component Analyzer application
Example	INP:POL:MUELL 0

4.2.4.8 INPut:POLarization:MUELLermode?

Syntax	**INPut:POLarization:MUELLermode?**
Description	Queries the state of the polarization analysis. Refer to the User Guide for further explaining.
Response	It returns '1' or 'ON' when is set to the advance polarization mode and '0' or 'OFF' otherwise.
Example	INP:POL:MUELL?

4.2.4.9 INPut:POWer?

Syntax	**INPut:POWer?**
Description	Queries total optical power being input to the BOSA. It can be also used in Option 210 for a readout of the Aux Input photodetector.
Example	INP:POW?

4.2.5 CALCulate Subsystem Commands

4.2.5.1 CALCulate:MARKer:AOFF

Syntax	**CALCulate:MARKer:AOFF**
Description	Disable the marker.
Example	CALC:MARK:AOFF
Comments	Referred to the active trace.

4.2.5.2 CALCulate:MARKer:STATe

Syntax	**CALCulate:MARKer:STATe ON|OFF|1|0**
Description	Enables or disables the marker.
Argument	'on' or '1' to turn the marker on, or 'off' or '0' to turn it off.
Example	CALC:MARK:STAT ON
Comments	Referred to the active trace.

4.2.5.3 CALCulate:MARKer:STATe?

Syntax	**CALCulate:MARKer:STATe?**
Description	Queries the state of the marker.
Response	The current state of the marker: 'ON' or 'OFF.
Example	CALC:MARK:STAT?
Comments	Referred to the active trace.

4.2.5.4 CALCulate:MARKer:MODe

Syntax	**CALCulate:MARKer:MODe TRCK|FIXX|FIXXY**
Description	Sets the behaviour of the marker (and its offset if enabled).
Argument	The desired behaviour of the marker: 'TRCK', 'FIXX' or 'FIXXY'.
Example	CALC:MARK:MOD FIXX
Comments	Referred to the active trace.

4.2.5.5 CALCulate:MARKer:MODe?

Syntax	**CALCulate:MARKer:MODe?**
Description	Queries the behaviour of the marker.
Response	The current behaviour of the marker: 'TRCK', 'FIXX' or 'FIXXY'
Example	CALC:MARK:MOD?
Comments	Referred to the active trace.

4.2.5.6 CALCulate:MARKer:MAXimum

Syntax	**CALCulate:MARKer:MAXimum**
Description	Moves the specified marker to the maximum power level.
Example	CALC:MARK:MAX
Comments	Referred to the active trace.

4.2.5.7 CALCulate:MARKer:MAXimum:NEXT

Syntax	**CALCulate:MARKer:MAXimum:NEXT**
Description	Moves the marker to the next power maximum.
Example	CALC:MARK:MAX:NEXT
Comments	Referred to the active trace.

4.2.5.8 CALCulate:MARKer:MAXimum:RIGHT

Syntax	**CALCulate:MARKer:MAXimum:RIGHT**
Description	Moves the marker to the next power maximum to the right.
Example	CALC:MARK:MAX:RIGHT
Comments	Referred to the active trace.

4.2.5.9 CALCulate:MARKer:MAXimum:LEFT

Syntax	**CALCulate:MARKer:MAXimum:LEFT**
Description	Moves the marker to the next power maximum to the left.
Example	CALC:MARK:MAX:LEFT
Comments	Referred to the active trace.

4.2.5.10 CALCulate:MARKer:SCEN

Syntax	**CALCulate:MARKer:SCEN**
Description	Moves the marker to the center of the screen.
Example	CALC:MARK:SCEN
Comments	Referred to the active trace.

4.2.5.11 CALCulate:MARKer:X

Syntax	**CALCulate:MARKer:X <numeric_value> [NM|PM|GHZ|THZ]**
Description	Moves the marker to the specified point.
Argument	'numeric_value' is the horizontal coordinate to put the marker on.
Example	CALC:MARK:X 1550 NM
Comments	Referred to the active trace.

4.2.5.12 CALCulate:MARKer:X?

Syntax	**CALCulate:MARKer:X?**
Description	Queries the horizontal coordinate of the marker.
Response	The horizontal coordinate of the specified marker, expressed in units set by the READOUT function.
Example	CALC:MARK:X?
Comments	Referred to the active trace.

4.2.5.13 CALCulate:MARKer:Y

Syntax	**CALCulate:MARKer:Y <numeric_value> [DBM|MW]**
Description	Moves the marker to the specified point on the vertical axis.
Argument	'numeric_value' is the vertical coordinate to put the marker on.
Example	CALC:MARK:Y -20 DBM
Comments	Referred to the active trace.

4.2.5.14 CALCulate:MARKer:Y?

Syntax	**CALCulate:MARKer:Y?**
Description	Queries the vertical coordinate of the marker.
Response	The vertical coordinate of the marker, expressed in the units set by the READOUT function.
Example	CALC:MARK:Y?
Comments	Referred to the active trace.

4.2.5.15 CALCulate:MARKer:THREshold

Syntax	**CALCulate:MARKer:THREshold <numeric_value> [DB]**
Description	Sets the minimum power level for the search of peaks in peak search function
Argument	'numeric_value' is the depth of the peak search configuration.
Example	CALC:MARK:THRES-3 DB

4.2.5.16 CALCulate:MARKer:THREshold?

Syntax	**CALCulate:MARKer:THREshold?**
Description	Queries the current peak excursion value for the peak search functions.
Response	The value for the peak threshold.
Example	CALC:MARK:THRES?

4.2.5.17 CALCulate:MARKer:READout

Syntax	**CALCulate:MARKer:READout FREQuency|WAVelenght**
Description	Sets the horizontal units of the marker
Example	CALC:MARK:READ FREQ

4.2.5.18 CALCulate:MARKer:READout?

Syntax	**CALCulate:MARKer:READout?**
Description	Queries the horizontal units of the marker
Example	CALC:MARK:READ?

4.2.5.19 CALCulate:MARKer:SRLevel

Syntax	**CALCulate:MARKer:SRLevel**
Description	Moves the reference level to the vertical value of the marker.
Example	CALC:MARK:SRL
Comments	Referred to the active trace. Activate marker on desired trace first, otherwise the command will return error.

4.2.5.20 CALCulate:MARKer:POLarization?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:POLarization?**
Description	In polarimetry extension, queries the S parameters (S1, S2, S3) of the marker in ASCII format.
Example	CALC:MARK:POL?
Comments	Values separated by commas.

4.2.5.21 CALCulate:MARKer:FUNCtion:DELTa

Syntax	**CALCulate:MARKer:FUNCtion:DELTa [:STATE] ON|OFF|1|0**
Description	Turns the delta marker function on or off.
Example	CALC:MARK:FUNC:DELTA ON
Comments	Referred to the active trace.

4.2.5.22 CALCulate:MARKer :FUNCtion:DELTa?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa [:STATE]?**
Description	Queries the state of the delta marker function.
Example	CALC:MARK:FUNC:DELT?
Comments	Referred to the active trace.

4.2.5.23 CALCulate:MARKer:FUNCtion:DELTa:RESet

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:RESet**
Description	Sets the marker offset to the current position of the marker.

Example	CALC:MARK:MARK:FUNC:DELT:RES
Comments	Referred to the active trace.

4.2.5.24 CALCulate:MARKer:FUNCtion:DELTa:X:OFFSet?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:X:OFFSet?**
Description	Queries the horizontal offset between the marker and its reference.
Example	CALC:MARK:FUNC:DELT:X:OFFS?
Comments	Referred to the active trace.

4.2.5.25 CALCulate:MARKer:FUNCtion:DELTa:X:REFerence?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:X:REFerence?**
Description	Queries the horizontal value of the reference marker.
Example	CALC:MARK:FUNC:DELT:X:REF?
Comments	Referred to the active trace.

4.2.5.26 CALCulate:MARKer:FUNCtion:DELTa:Y:OFFSet?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:Y:OFFSet?**
Description	Queries the vertical offset between the marker and its reference.
Example	CALC:MARK:FUNC:DELT:Y:OFFS?
Comments	Referred to the active trace.

4.2.5.27 CALCulate:MARKer:FUNCtion:DELTa:Y:REFerence?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:Y:REFerence?**
Description	Queries the vertical value of the reference marker.
Example	CALC:MARK:FUNC:DELT:Y:REF?
Comments	Referred to the active trace.

4.2.5.28 CALCulate:MARKer:FUNCtion:DELTa:POL?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:POLarization?**
Description	In polarimetry extension, queries the S parameters (S1, S2, S3) of the offset in ASCII format.
Example	CALC:MARK:FUNC:DELT:POL?
Comments	Values separated by commas.

4.2.5.29 CALCulate:MARKer:FUNCtion:DELTa:ANG?

Syntax	**CALCulate:MARKer:FUNCtion:DELTa:ANGle?**
Description	In polarimetry extension, queries the angle between marker and offset in ASCII format.
Example	CALC:MARK:FUNC:DELT:ANG?
Comments	

4.2.5.30 CALCulate:MAXimum[:STATe]

Syntax	**CALCulate:MAXimum[:STATe] 0|1|ON|OFF**
Description	Enables or disables the Max Hold function.
Argument	'1' or 'ON' to enable the Max Hold function; '0' or 'OFF' to disable it.
Example	CALC:MAX ON

4.2.5.31 CALCulate:MAXimum[:STATe]?

Syntax	**CALCulate:MAXimum[:STATe]?**
Description	Queries the Max Hold function state.
Response	'ON' if it is enabled, 'OFF' if not.
Example	CALC:MAX?

4.2.5.32 CALCulate:MINimum[:STATe]

Syntax	**CALCulate:MINimum[:STATe] 0|1|ON|OFF**
Description	Enbles or disables the Min Hold function.
Argument	'1' or 'ON' to enable the Min Hold function; '0' or 'OFF' to disable it.
Example	CALC:MIN ON

4.2.5.33 CALCulate:MINimum[:STATe]?

Syntax	**CALCulate:MINimum[:STATe]?**
Description	Queries the Min Hold function state.
Response	'ON' if it is enabled, 'OFF' if not.
Example	CALC:MIN?

4.2.5.34 CALCulate:TPOWer ON/OFF

Syntax	**CALCulate:TPOWer 0|1|ON|OFF**
Description	Enables or disables the power integral function.
Argument	‘1’ or ‘ON’ to enable the Power Integral function; ‘0’ or ‘OFF’ to disable it.
Example	CALC:TPOW ON

4.2.5.35 CALCulate:TPOWer:[DATA]?

Syntax	**CALCulate:TPOWer:[DATA]?**
Description	Returns the power integral in mW or dBm.
Example	CALC:TPOW?

4.2.5.36 CALCulate:TPOWer:IRANge:UPPer

Syntax	**CALCulate:TPOWer:IRANge:UPPer<numeric_value> [nm|pm|THz|GHZ]**
Description	Sets the upper bound for power integration.
Argument	‘numeric_value’ is the value for the upper bound. Units are optional.
Example	CALC:TPOW:IRAN:UPP 1555 NM

4.2.5.37 CALCulate:TPOWer:IRANge:UPPer?

Syntax	**CALCulate:TPOWer:IRANge:UPPer?**
Description	Returns the upper bound for power integration in nm or GHz.
Example	CALC:TPOW:IRAN:UPP?

4.2.5.38 CALCulate:TPOWer:IRANge:LOWer

Syntax	**CALCulate:TPOWer:IRANge:LOWer <numeric_value> [nm|pm|THz|GHZ]**
Description	Sets the lower bound for power integration.
Argument	‘numeric_value’ is the value for the lower bound. Units are optional.
Example	CALC:TPOW:IRAN:LOW 1550 NM

Syntax	**CALCulate:TPOWer:IRANge:LOWer?**

Description	Returns the lower bound for power integration in nm or GHz.
Example	CALC:TPOW:IRAN:LOW?

4.2.5.39 CALCulate:TPOWer:IRANge:LOWer?

4.2.5.40 CALCulate:AUXINput:POWer?

Syntax	**CALCulate:TPOWer:IRANge:LOWer?**
Description	Returns the power meter measurement from the Aux Input port in dBm. Only available in the tunable laser aplication.
Example	CALC:AUXIN:POW?

4.2.6 TRACe Subsystem Commands

4.2.6.1 TRACe[:DATA]:COUNT?

Syntax	**TRACe[:DATA]:COUNT?**
Description	Queries the number of points displayed on the screen.
Response	The number of points showed in the BOSA screen. Each point represent a 64 bit numeric value.
Example	TRAC:COUNT?

4.2.6.2 TRACE[:DATA]?

Syntax	**TRACe[:DATA]?**
Description	Queries the whole spectrum measured by the BOSA.
Response	The entire trace displayed in the BOSA screen. Use FORMAT[:DATA] to specify the format of the returned data. When format is set to REAL points are 64 bit in length. Use TRACE:COUNT? to know the number of points returned. When format is set to ASCII the values are separated by commas, the whole returned string finishes with "\r\n".

Comments	This command only works when the BOSA is controlled remotely.
Example	TRAC?

4.2.6.3 TRACe[:DATA]:MAXimum:X?

Syntax	**TRACe[:DATA]:MAXimum:X?**
Description	Queries the X coordinate of the absolute maximum.
Response	The X coordinate of the absolute maximum.
Example	TRAC:MAX:X?

4.2.6.4 TRACe[:DATA]:MAXimum:Y?

Syntax	**TRACe[:DATA]:MAXimum:Y?**
Description	Queries the Y coordinate of the absolute maximum.
Response	The Y coordinate of the absolute maximum.
Example	TRAC:MAX:Y?

4.2.7 FORMat Subsystem Commands

4.2.7.1 FORMat[:DATA]

Syntax	**FORMat[:DATA] <ASCII|REAL>,<length>**
Description	Sets the output data format for TRACe[:DATA][:Y]?. Available formats are either to ASCII with a given length or REAL (64 bits).
Argument	'ASCII' or 'REAL' for the format and the 'length' of the string in case of the ASCII option.
Example	FORM ASCII,5

4.2.7.2 FORMat[:DATA]?

Syntax	**FORMat[:DATA]?**
Description	Returns the current output data format for TRACe[:DATA][:Y]?.
Response	Example: 'ASCII,6' or 'REAL,6'
Example	FORM?

4.2.8 MMEMory Subsystem Commands

4.2.8.1 MMEMory:STORe:TRACe

Syntax	**MMEMory:STORe:TRACe <name>.<type>,[ink_saving_mode]**
Description	Stores the specified trace in a file in the specified format.
Arguments:	'name' is the path of the new file (without the extension) and 'type' can be: bdf, txt, csv, jpg, bmp, gif or tif. Finally, 'ink_saving_mode' is an optional Boolean (1, 0, ON or OFF) parameter only for image formats. If it is true the image will be saved with ink saving mode colors. If it is omitted, the image will display the original BOSA colors set up. NOTE: the root directory is 'User Files'. The file name is relative to the root directory.
Example	MMEM:STOR:TRAC myImages\image1.jpg,1

4.2.8.2 MMEMory:DELete:TRACe

Syntax	**MMEMory:DELete:TRACe <name>**
Description	Deletes the specified file
Arguments:	'name' is the file name to be deleted. The root directory is 'User Files'. The file name is relative to the root directory.
Example	MMEM:DEL:TRAC gpr.csv

4.2.8.3 MMEMory:LOAD:TRACe

Syntax	**MMEMory:LOAD:TRACe <M1|M2|M3|M4>,<name>**
Description	Loads the specified file
Arguments:	'M1' to 'M4' sets the trace in which the BOSA software will load the file, 'name' is the file name to be loaded. The root directory is 'User Files'. The file name is relative to the root directory.
Example	MMEM:LOAD:TRAC M2,gpr.csv

5 Programming examples

5.1 Ethernet Interface

The following example of C code shows how to communicate with the BOSA using the Ethernet Interface. A query is sent to the BOSA and its response is read.

```
// SocketsCExample.cpp : Uses the ethernet communication
// open a socket, send *IDN? query and print the response
#include "stdafx.h"

// include socket library
#include <winsock2.h>
#pragma comment(lib, "ws2_32.lib")

#define LEN_REP 255 //length of the response buffer

int main(int argc, char* argv[])
{
	WSADATA WSAData;
	SOCKET sock;
	SOCKADDR_IN sin;
	char response[LEN_REP];
	int byte_read;
	int i;

	//initialize WINSOCK
	WSAStartup(MAKEWORD(2,0), &WSAData);

	//Take the address of the instrument
	sin.sin_addr.s_addr = inet_addr("192.168.127.79");
	sin.sin_family = AF_INET;
	sin.sin_port = htons(10000);

	if((sock=socket(sin.sin_family,SOCK_STREAM,0)) < 0)
	{
		perror("Creation socket failed");
		return(1);
	}

	if(connect(sock,(SOCKADDR *)&sin, sizeof(sin))<0)
	{
		perror("Connect failed");
		return(1);
	}
```

```
    //send the command
    send(sock, "*IDN?", strlen("*IDN?"), 0);

    //-1: leave space for null terminator

    if((byte_read=recv(sock, response, LEN_REP-1, 0))<=0)
    {
          perror("Receive error");
          return(1);
    }
    response[byte_read] = 0x00; // terminate the string

    printf("%s",response);
    closesocket(sock);
    WSACleanup();


    return 0;
```

5.2 Macro Editor Tool

5.2.1 Performing auto measurements of several peaks using markers

The following example shows how to use markers to perform automatic searching and measurements of several peaks in the spectrum.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                  MULTIPEAK MEASUREMENT                                     //
////////////////////////////////////////////////////////////////////////////////////////////////////////////

//      DESCRIPTION

//      This macro will search for peaks present in the configured span,
//      save their wavelength and power in an array and finally store the
//      array values in a file.

//      DECLARATION AND INITIALIZATION OF VARIABLES

CENTERWL = 1550                                          // The center wavelength in nm
SPAN = 5                                                 // The desired span, in nm
REFLEVEL = -50                                           // The Reference level
DATE = FORMATDATE(NOW(),"MMMM_dd hh_mm")                 // The actual date
FILE = "PeaksMeasurement\results " & DATE & ".txt"       // Results will be saved in this file
PEAKPOSITION = ARRAY(20)                                 // It will contain the peak positions
PEAKPOWER = ARRAY(20)                                    // It will contain the peak powers

COUNT=0
```

```
//        INSTRUCTIONS

//        Configuration of the BOSA: Center and span
SCPI("SENS:WAV:CENT" & " " & CENTERWL & " " & "NM")
SCPI("SENS:WAV:SPAN" & " " & SPAN & " " & "NM")

//        Configuration of the BOSA: Peak Search parameters
//        Markers will only find peaks with a peak excursion greater than
//        'pexcursion' at a distance of 'pwidth' from the wavelength of the
//        maximum of the peak
SCPI("CALC:MARK1:PEXCURSION 10 DB")
SCPI("CALC:MARK1:PWIDTH 0.0002 NM")

//        Configuration of the BOSA: Reference Level.
//        The reference level is used as threshold for peak search.
//        Peaks with amplitude lower than reflevel will not be found
SCPI("DISP:TRAC:Y:RLEV" & " " & REFLEVEL & " " & "DBM")

//        Activate the marker 1
SCPI("CALC:MARK1:STAT ON")
//        Move the marker 1 to the starting wavelength
SCPI("CALC:MARK1:X STAR")
//        Move the marker 1 to the first peak
SCPI("CALC:MARK1:MAX:NEXT")
WAIT(200)

//        Ask the BOSA the peak parameters
POSITION = TODOUBLE(SCPI("CALC:MARK1:X?"))
POWER = TODOUBLE(SCPI("CALC:MARK1:Y?"))


//        MEASUREMENT LOOP

REPEAT
          PEAKPOSITION[COUNT] = POSITION
          PEAKPOWER[COUNT] = POWER
          COUNT = COUNT+1
          SCPI("CALC:MARK1:MAX:NEXT")
          WAIT(200)
          POSITION = TODOUBLE(SCPI("CALC:MARK1:X?"))
POWER = TODOUBLE(SCPI("CALC:MARK1:Y?"))
WHILE(POSITION<PEAKPOSITION[COUNT-1]&&COUNT<LENGTH(PEAKPOSITION))

TRIM(PEAKPOSITION)
TRIM (PEAKPOWER)

//        SAVING DATA TO A FILE

NUMFORMAT = "0.0000"
FWRITELN(FILE,"SAMPLE MACRO FOR MULTIPLE PEAKS AUTOMATED MEASUREMENT")
FWRITELN(FILE,"")
FWRITELN(FILE,"PEAK#   WL(NM)   POW(DBM)")
j=0
REPEAT
          FWRITE(FILE,TOSTRING(J+1))
          FWRITE(FILE,"       " & FORMATNUM(PEAKPOSITION[J] , NUMFORMAT ))
          FWRITELN(FILE,"  " & FORMATNUM(PEAKPOWER[J] , NUMFORMAT ))
          J=J+1
WHILE(J<LENGTH("PEAKPOWER"))
```

5.2.2 Wavelength shift measurement on a CW laser

The following example shows how to perform an automatic measurement of the wavelength shift on a CW laser over time.

```
/////////////////////////////////////////////////////////////////////////////////////////////////////////////
//        WAVELENGTH SHIFT MEASUREMENT ON A CW LASER        //
/////////////////////////////////////////////////////////////////////////////////////////////////////////////

//        DESCRIPTION

//        The following program measures the wavelenght emition of a CW  laser
//        over time. A wavelenght measurement is taken and stored in a data
//        array at the specified time interval, finally the results are plot in a chart.


//        DECLARATION AND INITIALIZATION OF VARIABLES

NUMMEAS=100                       // Number of measurements to be taken
TIMEINTERVAL=10                   // Time interval between measurements in seconds
INDEX=0

DATA=ARRAY(NUMMEAS)

//        INSTRUCTIONS

//        An auto-measurement is performed to locate the output laser wavelength
SCPI("DISP:TRAC:X:AUTO")
WAIT(20000)



//        MEASUREMENT LOOP

REPEAT
        //            It queries the wavelength of the power maximum
        WMAX=SCPI("TRACE:DATA:MAXIMUM:X?")
        //            The wavelength is store in the data array
        DATA[INDEX]=TODOUBLE(WMAX)
       //             The measurement center wavelength is set to the wavelength of the
       //              measured maximum. It avoids the laser to shift out the measurement span.
        SCPI("SENS:WAV:CENTER " & WMAX & " NM")
        //            The time interval between measurements is used in milliseconds
        WAIT(TIMEINTERVAL*1000)
        INDEX=INDEX+1
WHILE (INDEX<NUMMEAS)

//        The DATA array is plot in a chart
PLOT(DATA)

SORT(DATA)
MAXIMUM=DATA[NUMMEAS-1]
MINIMUM=DATA[0]
DELTA=MAXIMUM-MINIMUM

SCPI("INST:STAT OFF")

//            It is possible to reach the numeric data by following the path File -> Save as Text on the
//            chart. A txt file is
 //           saved with all the numeric values plotted on the chart.
```

5.2.3 Controlling an external device via GPIB

The following example shows how to manage an external device through GPIB using its own SCPI commands. Note that these commands depend on each device and they could be different for each model.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                          MANAGING AN EXTERNAL TLS VIA GPIB                  //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//        DESCRIPTION

//        The following program measures the wavelength and the power of
//        an external TLS which is set to perform stepped sweeps.


//        DECLARATION AND INITIALIZATION OF VARIABLES

WSTART= 1540
WSTOP = 1560
INTERVAL = 2
POWER = 0
WLENGTH = 0
COUNT = 0
LAMBDA = 0
FILE = "externalTLS.txt"                 // Results will be saved in this file
IDTLS = 10                               // The GPIB ID number of the TLS
SLOTTLS = "0"                            // The slot number of the TLS???

WAVELENGTHTHEO = ARRAY(11)               // 11 = size = 1 + (wstop-wstart)/interval
WAVELENGTHREAL = ARRAY(11)
POWERREAL = ARRAY(11)


//        INSTRUCTIONS

//        Configuration of the BOSA: Start and stop
SCPI("SENS:WAV:START" & " " & (WSTART-INTERVAL)& " " & "NM")
SCPI("SENS:WAV:STOP" & " " & (WSTOP+INTERVAL) & " " & "NM")

//        The BOSA is configured to measure at the highest resolution
SCPI("SENS:WAV:RES HIGH")

LAMBDA = WSTART

//        Configuration of the external TLS: Lambda and power
SENDGPIB(IDTLS, "SOUR"&SLOTTLS&":WAV "& FORMATNUM(LAMBDA,"0.###")&" NM")
SENDGPIB(IDTLS, "SOUR"&SLOTTLS&":POW "& FORMATNUM(POWER,"0.0")&" DBM")
//        Turn on the external TLS
SENDGPIB(IDTLS, "SOUR"&SLOTTLS&":POW:STAT 1")

WAIT(7500)

//        It queries the wavelength of the maximum
WLENGTH = TODOUBLE(SCPI("TRACE:DATA:MAXIMUM:X?"))
//        It queries the power of the maximum
POWER = TODOUBLE(SCPI("TRACE:DATA:MAXIMUM:Y?"))


//        MEASUREMENT LOOP

REPEAT
        WAVELENGTHTHEO[COUNT] = LAMBDA
        WAVELENGTHREAL[COUNT] = WLENGTH
        POWERREAL[COUNT] = POWER
        COUNT = COUNT+1

        LAMBDA = LAMBDA +INTERVAL
```

```
          //          Configuring the external TLS: Lambda
          SENDGPIB(IDTLS, "SOUR"&SLOTTLS&":WAV "& FORMATNUM(LAMBDA,"0.###")&" NM")

          WAIT(5000)

          //          It queries the wavelength of the maximum
          WLENGTH = TODOUBLE(SCPI("TRACE:DATA:MAXIMUM:X?"))
          //          It queries the the power of the maximum
          POWER = TODOUBLE(SCPI("TRACE:DATA:MAXIMUM:Y?"))
WHILE (LAMBDA<=WSTOP)

//        Turn off the external TLS
SENDGPIB(IDTLS, "SOUR"&SLOTTLS&":POW:STAT 0")


//        SAVING DATA TO A FILE

NUMFORMAT = "0.0000"
FWRITELN(FILE,"SAMPLE MACRO FOR MANAGING EXTERNAL TLS")
FWRITELN(FILE,"")
FWRITELN(FILE,"WL THEO (NM)    WL REAL (NM)      POWER(DBM)")
J=0
REPEAT
          FWRITE(FILE,"      " & FORMATNUM(WAVELENGTHTHEO[J] , NUMFORMAT ))
          FWRITE(FILE,"      " & FORMATNUM(WAVELENGTHREAL[J] , NUMFORMAT ))
          FWRITELN(FILE,"  " & FORMATNUM(POWERREAL[J] , NUMFORMAT ))
          J=J+1
WHILE(J<LENGTH(POWERREAL ))
```