3680A/B Cable & Antenna Analyzer

Programming Manual

China Electronics Technology Instruments Co., Ltd

July, 2016

PREFACE

Thank you for choosing and using 3680A/B Cable & Antenna Analyzer developed and produced by China Electronics Technology Instruments Co., Ltd. Please read this manual carefully for the convenience of operation.

We will assume trying our best to meet your needs as our responsibility to provide high quality instruments for you and also bring you first-class after-sale service. We always persist in “Good Quality, Satisfied Service” and promise to offer satisfactory products and service for you. Welcome to inquire by:

Tel:	**+86-0532-86896691**
Website:	**www.ceyear1.com**
Email:	**sales@ceyear.com**
Address:	**No.98, Xiangjiang Road, Qingdao City, China**
Zip	**266555**

This document provides information about the implementation of program control functions and the program control commands for 3680A/B Cable & Antenna Analyzer produced by China Electronics Technology Instruments Co., Ltd. Please read this manual carefully and operate the instrument correctly according to the instructions herein.

Because of short time and the limited knowledge, some mistakes and shortcomings are inevitable. Welcome your criticism and correction. And apologize for your possible inconvenience when you use our instruments.

☛ **STATEMENT:**

This is the first version of programming manual of 3680A/B Cable & Antenna Analyzer, and the version number is A AV2.733.1040SCCN.

The information contained in this manual is subject to change without notice. And the explanation right of all the content and the terms in this manual belongs to China Electronics Technology Instruments Co., Ltd.

The copyright of the manual belongs to China Electronics Technology Instruments Co., Ltd. Other entities and individuals can’t change or juggle it without our permission, and can’t copy and spread for earn business benefits. If discovered, we have the right for the legal action.

Editor

July, 2016

CONTENTS

Chapter 1 Overview

3680A/B Cable & Antenna Analyzer is an affordable and superior-performance measuring instrument capable of conducting a comprehensive measurement of network reflection parameters. In addition to measuring SWR, Return Loss, Characteristic Impedance and Phase, the 3680A/B also can determine exact cable defect locations. Furthermore, the power measurement mode is also available with option, in which, precise power measurement can be implemented through an external power sensor.

SCPI (Standard Commands for Programmable Instruments) is a new command language for programmable instruments in accordance with IEEE488.2 standard. It mainly aims to make similar instruments have the same program control commands to standardize the program control command. 3680A/B Cable & Antenna Analyzer uses such SCPI. The communication between the tester and external control computer can be established via the tester's USB or LAN interface, so that remote control can be implemented by way of sending identifiable SCPI commands to the tester. In addition, the SCPI commands are packaged as a dynamic library function, the programmable functionality can be quickly implemented by use of the secondary development libraries.

Chapter 2 covers the information of all SCPI commands identified and executed by 3680A/B, contained IEEE488.2 common commands and instrument specific control commands. The instrument specific control commands are introduced in three sections, i.e. specific commands for cable and antenna measurement mode, specific commands for power measurement mode and universal commands for all the modes. In each section, the commands are further subdivided in terms of SCPI command subsystem.

Chapter 3 covers the introduction of the use of secondary development libraries and the functions contained therein. The part of function introduction is divided into five sections, i.e. instrument connecting commands, SCPI common commands, specific commands for antenna measurement mode, specific commands for power measurement mode and universal commands for all the modes. And these specific commands are introduced further by menu.

Chapter 2 SCPI Command Description

This Chapter covers the information of all SCPI commands identified and executed by 3680A/B, IEEE488.2 common commands and instrument specific control commands. The instrument specific control commands are introduced in three sections, i.e. specific commands for cable and antenna measurement mode, specific commands for power measurement mode and universal commands. Universal commands include those universal in two modes, i.e. file operation and system operation.

2.1 IEEE 488.2 Common Commands

*CLS - Clear Status

Clear the instrument status by emptying the error queue and clearing all event registers, and cancel all pending *OPC commands and query commands.

*IDN? - Identification

Return a string that uniquely identifies the instrument. Different models have different identifications. For example: “CETC41, 3680A/B, SN,1.00”.

*OPC - Operation Complete Command

Set the OPC position of standard event status register after all the pending overlapped commands have been completed (for example: a sweep or Default Command, etc.).

*OPC? - Operation Complete Query

Return “1” when all pending overlapped commands have been completed.

*RST - Reset

Execute a device reset and cancel any pending *OPC command or query, with the same functionality as the command SYSTem:PRESet.

*WAI - Wait

Prohibit the instrument from executing any new commands until all pending overlapped commands have been completed.

2.2 Specific Commands for Cable & Antenna Testing

2.2.1 :CALCulate Subsystem

:CALCulate:FORMat <string>

(Read and Write) Set or query the current measurement format.

Parameter Measurement format

Measurement format	Set parameter <string>	Return value <int>
Return Loss	RLOSs	0
SWR	VSWR	1
Cable Loss	CLOSs	2
DTF SWR	DTFV	3
DTF Return Loss	DTFR	4
Smith Chart	SMITh	5
Phase Diagram	PHASe	6

Example :CALC:FORM CLOS

Query Syntax :CALC:FORM?

Default RLOss

Return Type Numeric (int)

Menu Operation 【Meas】

:CALCulate:LIMit:TEST<ON | OFF>

(Read and Write) Set or query the ON or OFF of limit test.

Parameter Limit test On/Off

Limit test On/Off	Set parameter <ON | OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :CALC:LIM:TEST ON

Query Syntax :CALC:LIM:TEST?

Default OFF

Return Type Numeric (int)

Menu Operation 【Limit】－［Limit Off On］

:CALCulate:LIMit:FAIL

(Read Only) Query the result of limit test.

Parameter Not Applicable

Example :CALCulate:LIMit:FAIL?

Query Syntax :CALCulate:LIMit:FAIL?

Default Not Applicable

Return Type Numeric (int)

0: limit test succeeded.

1: limit test failed.

Menu Operation Not Applicable

:CALCulate:LIMit:BEEP <ON | OFF>

(Read and Write) Set or query the ON or OFF of limit test beep.

Parameter Limit test beep On/Off

Limit test beep On/Off	Set parameter <ON | OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :CALC:LIM:BEEP ON

Query Syntax :CALC:LIM:BEEP?

Default OFF

Return Type Numeric (int)

Menu Operation 【Limit】 — ［Beep Off On］

:CALCulate:LIMit:MODE <string>

(Read and Write) Set or query the limit line mode.

Parameter

Limit mode	Set parameter <string>	Return value <int>
Upper	UPPER	1
Lower	LOWER	0

Example :CALC:LIM:MODE UPPER

Query Syntax :CALC:LIM:MODE?

Default UPPER

Return Type Numeric (int)

Menu Operation 【Limit】 — ［Mode Upper Lower］

:CALCulate:LIMit:EDIT:ADD

(Write Only) Add limit point. Add a new limit point in between the current limit point and the next limit point during edit of limit line.

Parameter Not Applicable

Example :CALC:LIM:EDIT:ADD

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Limit】 — ［Edit］ — ［Add］

:CALCulate:LIMit:EDIT:CLEar

(Write Only) Clear all the limit points.

Parameter Not Applicable

Example :CALC:LIM:EDIT:CLE

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Limit】－［Edit］－［Clear］

:CALCulate:LIMit:EDIT:DELete

(Write Only) Delete the current limit point.

Parameter Not Applicable

Example :CALC:LIM:EDIT:DEL

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Limit】－［Edit］－［Delete］

:CALCulate:LIMit:EDIT:SELect <num>

(Read and Write) Set or query the current limit point. The number of limit point starts from 1 and increases in order from left to right.

Parameter Number of limit point.

Example :CALC:LIM:EDIT:SEL 5

Query Syntax :CALC:LIM:EDIT:SEL?

Default Not Applicable

Return Type Numeric (int)

Menu Operation 【Limit】－［Edit］－［Limit Point］

:CALCulate:LIMit:EDIT:X <num>

(Read and Write) Set or query the X-axis number of current limit point. In non DTF measurement, it is the frequency value in Hz; in DTF measurement, it is the distance value in current DTF unit.

Parameter X-axis number of limit point.

Example

1. In non DTF measurement, set the frequency of limit point to 5MHz:
   :CALC:LIM:EDIT:X 5000000
2. In DTF measurement, the distance unit is “m”, and set the location of limit point to 5m:
   :CALC:LIM:EDIT:X 5

Query Syntax :CALC:LIM:EDIT:X?

Default Not Applicable

Return Type Numeric (double)

Menu Operation 【Limit】－［Edit］－［Freq］/［Dist］

:CALCulate:LIMit:EDIT:Y <num>

(Read and Write) Set or query the amplitude of current limit point, which is equal to the unit of current Y-axis.

Parameter Amplitude of limit point.

Example :CALC:LIM:EDIT:Y 50

Query Syntax :CALC:LIM:EDIT:Y?

Default Not Applicable

Return Type Numeric (double)

Menu Operation 【Limit】－［Edit］－［Limit Value］

:CALCulate:LIMit:POINts

(Read Only) Query the number of limit points.

Parameter Not Applicable

Example :CALC:LIM:POIN?

Query Syntax :CALC:LIM:POIN?

Default Not Applicable

Return Type Numeric (int)

Menu Operation 【Limit】－［Edit］

:CALCulate:MARKer[X][:STATe] <string>

(Read and Write) Set or query the state of specified marker. In the command, [X] is the index number of marker that is an integer number from 1 to 6. “Off” means both the basic reference marker and the delta marker are off. “Normal” means the basic reference marker is active, but the delta marker is off. “Delta” means both the basic reference marker and the delta marker are active

Parameter

Marker state	Set parameter <string>	Return value <int >
Off	OFF	0
Normal	NORMAL	1
Delta	DELTa	2

Example :CALC:MARK1 DELTa

Query Syntax :CALC:MARK1?

Default OFF

Return Type Numeric (int)

Menu Operation 【Marker】－［Marker 1 2 3 4 5 6］/［Mode Normal Delta］

:CALCulate:MARKer:AOFF

(Write Only) Turn off all the markers

Parameter	Not Applicable
Example	:CALC:MARK[X]:AOFF
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Marker】－［All Off］

:CALCulate:MARKer[X]:FUNCtion:MAXimum

(Write Only) Set the specified marker to the maximum. In the command, [X] is the index number of marker that is an integer number from 1 to 6.

Parameter	Not Applicable
Example	:CALC:MARK1:FUNC:MAX
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Marker】－［Peak］

:CALCulate:MARKer[X]:FUNCtion:MINimum

(Write Only) Set the specified marker to the minimum. In the command, [X] is the index number of marker that is an integer number from 1 to 6.

Parameter	Not Applicable
Example	:CALC:MARK1:FUNC:MIN
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Marker】－［Valley］

:CALCulate:MARKer[X]:X <num>

(Read and Write) Set or query the X value of specified marker. In the command, [X] is the index number of marker that is an integer number from 1 to 6. In non DTF measurement, it is the frequency in Hz; in DTF measurement, it is the distance in current DTF unit.

Parameter	X value of the specified marker.
Example	:CALC:MARK1:X 500000
Query Syntax	:CALC:MARK1:X?
Default	Not Applicable
Return Type	Numeric (double)
Menu Operation	【Marker】

:CALCulate:MARKer[X]:Y

(Read Only) Set or query the Y value of specified marker. In the command, [X] is the index number of marker that is an integer number from 1 to 6. Y value is expressed in complex number and returns two real numbers of single precision, the forepart of which is real part and the afterpart is imaginary part. In non Smith chart, the imaginary part is 0; in Smith measurement, the complex number is returned.

Parameter Not Applicable

Example :CALC:MARK1:Y?

Query Syntax :CALC:MARK1:Y?

Default Not Applicable

Return Type Numeric (float array)

Menu Operation 【Marker】

:CALCulate:MATH:FUNCtion <string>

(Read and Write) Set or query the trace operation or display type.

Parameter

Trace operation type	Set parameter <string>	Return value <int>
Data	DATA	0
Memory	MEM	1
Data && Memory	AND	2
Data - Memory	SUB	3
Data/Memory	DIV	4

Example :CALC:MATH:FUNC DIV

Query Syntax :CALC:MATH:FUNC?

Default DATA

Return Type Numeric (int)

Menu Operation 【Trace】

:CALCulate:MATH:MEMorize

(Write Only) Memorize the current trace.

Parameter Not Applicable

Example :CALC:MATH:MEM

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Trace】－［Data--->Memory］

:CALCulate:SMOothing[:STATe] <ON | OFF>

(Read and Write) Set or query the ON or OFF of smoothing

Parameter Smoothing On/Off.

Smoothing ON/OFF	Set parameter <ON | OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :CALC:SMO ON

Query Syntax :CALC:SMO?

Default OFF

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［Avg/BW］－［Smoothing Off On］

:CALCulate:SMOothing:APERture <num>

(Read and Write) Set or query the percentage of smoothing aperture, the effective range of which is [0.01~20].

Parameter Percentage of smoothing aperture

Example :CALC:SMO:APER 10

Query Syntax :CALC:SMO:APER?

Default 10

Return Type Numeric (double)

Menu Operation 【Sweep/Setup】－［Avg/BW］－［Smoothing Aperture］

:CALCulate:TRANsform:CLOSs <num>

(Read and Write) Set or query the cable loss value in dB/m or dB/ft, depending on the current DTF unit, the effective range of which is [0~30] dB/m.

Parameter Cable loss value.

Example :CALC:TRAN:CLOS 0.01

Query Syntax :CALC:TRAN:CLOS?

Default 0

Return Type Numeric (double)

Menu Operation 【Freq/Dist】－［Cable Loss］

:CALCulate:TRANsform:VFACtor <num>

(Read and Write) Set or query the velocity factor in DTF measurement, the effective range of which is [0.001~1].

Parameter Velocity factor

Example :CALC:TRAN:VFAC 0.8

Query Syntax :CALC:TRAN:VFAC?

Default 1

Return Type Numeric (double)

Menu Operation 【Freq/Dist】－［Velocity］

:CALCulate:TRANsform:CABLe < num>

(Read and Write) Set or query the cable model number. In the setting command, the cable is referenced by the index location in the list of cable models. In the query command, cable name is returned.

Parameter Not Applicable

Example :CALC:TRAN:CABL 2

Query Syntax :CALC:TRAN:CABL?

Default Not Applicable

Return Type Character string (char[])

Menu Operation 【Freq/Dist】－［Cable］

:CALCulate:TRANsform:WINDow <string>

(Read and Write) Set or query DTF window function.

Parameter

Window function	Set parameter <string>	Return value <integer>
Rectangular window	RECT	0
Normal edge smoothing	HANNing	1
Low edge Smoothing	HAMMing	2
Minimum edge smoothing	BLACkman	3

Example :CALC:TRAN:WIND RECT

Query Syntax :CALC:TRAN:WIND?

Default RECT

Return Type Numeric (int)

Menu Operation 【Freq/Dist】－［Next］

:CALCulate:TRANsform:DISTance:STARt <num>

(Read and Write) Set or query the value of start distance in DTF measurement, in m or ft depending on the current setting of DTF.

Parameter Value of start distance, the maximum of which to be set is determined by the following formula:

DMax = Velocity of light × Velocity factor × (Sweep points － 1)/Span × 0.5.

Take as 2000m if it is larger than 2000 m.

Example :CALC:TRAN:DIST:STAR 1

Query Syntax :CALC:TRAN:DIST:STAR?

Default 0

Return Type Numeric (double)

Menu Operation 【Freq/Dist】－［Start Distance］

:CALCulate:TRANsform:DISTance:STOP <num>

(Read and Write) Set or query the value of stop distance in DTF measurement, in m or ft depending on the current setting of DTF.

Parameter Value of stop distance, the maximum of which to be set is determined by the following formula:

DMax = Velocity of light × Velocity factor × (Sweep points － 1)/ Span × 0.5.

Take as 2000m if it is larger than 2000 m.

Example :CALC:TRAN:DIST:STOP 1

Query Syntax :CALC:TRAN:DIST:STOP?

Default 0

Return Type Numeric (double)

Menu Operation 【Freq/Dist】－［Stop Distance］

:CALCulate:TRANsform:DISTance:UNIT <string>

(Read and Write) Set or query the unit of DTF.

Parameter DTF unit.

DTF unit	Set parameter <string>	Return value <integer>
Metric	METER	0
Imperial	FEET	1

Example :CALC:TRAN:DIST:UNIT METER

Query Syntax :CALC:TRAN:DIST:UNIT?

Default METER

Return Type Numeric (int)

Menu Operation 【Freq/Dist】－［DTF Other］－［Unit］

:CALCulate:DATA:FDATA

(Read Only) Query the current trace data. Such data is the result of original data of current measurement through processing such as format conversion, averaging and smoothing. In the measurement format of non Smith chart, the trace data point is a real number and an array composed of data points is returned. In the format of Smith chart, the data point is a complex number, and every two adjacent elements in the array of real number returned represent the real part and imaginary part of a trace data point, and the forepart is the real part and the afterpart is the imaginary part.

Parameter Not Applicable

Example :CALCulate:DATA:FDATA?

Query Syntax :CALCulate:DATA:FDATA?

Default Not Applicable

Return Type <float> array

Menu Operation Not Applicable

:CALCulate:DATA:FMEM

(Read Only) Query the memory trace data. Such data is the result of original data of memory trace through processing such as format conversion, averaging and smoothing. In the measurement format of non Smith chart, the trace data point is a real number and an array composed of data points is returned. In the format of Smith chart, the data point is a complex number, and every two adjacent elements in the array of real number returned represent the real part and imaginary part of a trace data point, and the forepart is the real part and the afterpart is the imaginary part.

Parameter	Not Applicable
Example	:CALC:DATA:FMEM?
Query Syntax	:CALC:DATA:FMEM?
Default	Not Applicable
Return Type	<float> array
Menu Operation	Not Applicable

:CALCulate:DATA:SDATA

(Read Only) Obtain the original data of current trace, without processing such as format conversion, averaging and smoothing. If the error correction is active, then such data is the data with error corrected. The data point is in complex form, and every two adjacent elements in the array of real number returned represent the real part and imaginary part of an original data point, and the forepart is the real part and the afterpart is the imaginary part.

Parameter	Not Applicable
Example	:CALC:DATA:SDATA?
Query Syntax	:CALC:DATA:SDATA?
Default	Not Applicable
Return Type	<float> array
Menu Operation	Not Applicable

:CALCulate:DATA:SMEM

(Read Only) Obtain the original data of memory trace, without processing such as format conversion, averaging and smoothing. If the error correction is active when storing the memory trace, then such data is the data with error corrected. The data point is in complex form, and every two adjacent elements in the array of real number returned represent the real part and imaginary part of an original data point, and the forepart is the real part and the afterpart is the imaginary part.

Parameter	Not Applicable
Example	:CALC:DATA:SMEM?
Query Syntax	:CALC:DATA:SMEM?
Default	Not Applicable
Return Type	<float> array
Menu Operation	Not Applicable

2.2.2 :DISPlay Subsystem

:DISPlay:WINDow:TRACe:Y:SCALe:AUTO

(Write Only) Set the amplitude range automatically.

Parameter	Not Applicable
Example	:DISP:WIND:TRAC:Y:SCAL:AUTO
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Ampt】－［Auto Scale］

:DISPlay:WINDow:TRACe:Y:SCALe:TOP

(Read and Write) Set or query the top amplitude value.

Parameter	Top amplitude value.
Example	:DISP:WIND:TRAC:Y:SCAL:TOP 5
Query Syntax	:DISP:WIND:TRAC:Y:SCAL:TOP?
Default	Not Applicable
Return Type	Numeric (double)
Menu Operation	【Ampt】－［Top］

:DISPlay:WINDow:TRACe:Y:SCALe:BOTTom

(Read and Write) Set or query the bottom amplitude value.

Parameter	Bottom amplitude value.
Example	:DISP:WIND:TRAC:Y:SCAL:BOTTom -1
Query Syntax	:DISP:WIND:TRAC:Y:SCAL:BOTT?
Default	Not Applicable
Return Type	Numeric (double)
Menu Operation	【AMPTD】－［Top］

:DISPlay:WINDow:TRACe:Y:SCALe:DEFault

(Write Only) Resume the Default amplitude range.

Parameter	Not Applicable
Example	:DISP:WIND:TRAC:Y:SCAL:DEFault
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Ampt】－［Default Scale］

:DISPlay:WINDow:DUAL <ON | OFF>

(Read and Write) Set or query the ON or OFF of dual window display.

Parameter

Dual wind On/Off	Set parameter <O | OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :DISP:WIND:DUAL ON

Query Syntax :DISP:WIND:DUAL?

Default OFF

Return Type Numeric (int)

Menu Operation 【Meas】－［Dual Window Off On］

:DISPlay:WINDow:ACTive <string>

(Read and Write) Set or query the current active window.

Parameter

Meaning	Set parameter <string>	Return value <in >
Top window ID	0	0
Bottom window ID	1	1

Example :DISP:WIND:ACT 1

Query Syntax :DISP:WIND:ACT?

Default 0

Return Type Numeric (int)

Menu Operation 【Meas】－［Active Top Bottom］

2.2.3 :INITiate Subsystem

:INITiate[:IMMediate]

(Write Only) Immediately trigger sweep.

Parameter Not Applicable

Example :INIT

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation Not Applicable

:INITiate:CONTinuous <ON | OFF>

(Read and Write) Set or query the current triggering mode.

Parameter

Triggering mode	Set parameter <ON | OFF>	Return value <int>

Single	OFF	0
Continuous	ON	1

Example :INIT:CONT ON

Query Syntax :INIT:CONT?

Default ON

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】—［Trigger Mode Single Cont］

:INITiate:HOLD <ON | OFF>

(Read and Write) Set or query the ON or OFF of sweep hold.

Parameter

Run/Hold	Set parameter <O ｜OFF>	Return value <int>
Run	OFF	0
Hold	ON	1

Example :INIT:HOLD OFF

Query Syntax :INIT:HOLD?

Default OFF

Return Type Numeric (int)

Menu Operation 【Run/Hold】

2.2.4 :SENSe Subsystem

[:SENSe]:AVERage:COUNt <num>

(Read and Write) Set or query the average factor.

Parameter Average factor, the range of which to be set is (2~1000).

Example :AVER:COUN 5

Query Syntax :AVER:COUN?

Default 16

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】—［Avg/BW］—［Avg Factor］

[:SENSe]:AVERage:STATe <ON | OFF>

(Read and Write) Set or query the ON or OFF of average.

Parameter Average On/Off.

Average On/Off	Set parameter <ON｜ FF>	Return value <int>
Off	OFF	0
On	ON	1

Example :AVER:STAT ON

Query Syntax :AVER:STAT?

Default OFF

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［AVG/BW］－［Average Off On］

[:SENSe]:AVERage:CLEar

(Write Only) Clear the average counting (active when average is ON)

Parameter Not Applicable

Example :AVER:CLE

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation Not Applicable

[:SENSe]:BANDwidth <num>

(Read and Write) Set or query the bandwidth of intermediate frequency.

Parameter Intermediate frequency in Hz, the effective value of which (1Hz~10kHz) is in step of 1-2-5.

Example :BAND 1000

Query Syntax :BAND?

Default 1000

Return Type Numeric (double)

Menu Operation 【Sweep/Setup】－［AVG/BW］－［IF BW］

[:SENSe]:BWID <num>

(Read and Write) Set or query the bandwidth of intermediate frequency, same as [:SENSe]:BANDwidth

[:SENSe]:CORRection[:STATe] <ON | OFF>

(Read and Write) Error correction On or Off

Parameter

Error correction On or Off	Set parameter <ON OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :CORR ON

Query Syntax :CORR?

Default OFF

Return Type Numeric (int)

Menu Operation 【CAL】－［Calib Off On］

[:SENSe]:CORRection:COLLect:ACQuire:LOAD

(Read and Write) Conduct measurement of “Load” calibration standard or query whether the “Load” measurement is completed.

Parameter	Not Applicable
Example	:CORR:COLL:ACQ:LOAD
Query Syntax	:CORR:COLL:ACQ:LOAD?
Default	Not Applicable
Return Type	Numeric (int), 0: not measured; 1: measured
Menu Operation	【Cal】－［Mech Cal］－［Load］

[:SENSe]:CORRection:COLLect:ACQuire:OPEN

(Read and Write) Conduct measurement of “Open” calibration standard or query whether the “Open” measurement is completed.

Parameter	Not Applicable
Example	:CORR:COLL:ACQ:OPEN
Query Syntax	:CORR:COLL:ACQ:OPEN?
Default	Not Applicable
Return Type	Numeric (int), 0: not measured; 1: measured
Menu Operation	【Cal】－［M Cal］－［Open］

[:SENSe]:CORRection:COLLect:ACQuire:SHORt

(Read and Write) Conduct measurement of “Short” calibration standard or query whether the “Short” measurement is completed.

Parameter	Not Applicable
Example	:CORR:COLL:ACQ:SHOR
Query Syntax	:CORR:COLL:ACQ:SHOR?
Default	Not Applicable
Return Type	Numeric (int), 0: not measured; 1: measured
Menu Operation	【Cal】－［M Cal］－［Short］

[:SENSe]:CORRection:ABORt

(Write Only) Abort the calibration process. After the start of mechanical calibration, if the four calibration steps: Open - Short - Load – Finish, are not done, and calibration is no longer required, it is needed to release the instrument memory resources with this command.

Parameter	Not Applicable
Example	:CORR:ABOR
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable

Menu Operation Not Applicable

[:SENSe]:CORRection:DONE

(Write Only) Complete the measurement of all calibration standards. If the measurement of three calibration standards, i.e. Open-Short-Load, has been done, this command can be used to initiate calculation of error coefficient and turn on error correction automatically.

Parameter Not Applicable

Example :CORR:DONE

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Cal】－［M Cal］－［Done］

[:SENSe]:CORRection:VALid

(Read Only) Query the validity of calibration coefficient.

Parameter Not Applicable

Example :CORR:VAL?

Query Syntax :CORR:VAL?

Default Not Applicable

Return Type Not Applicable

Menu Operation Not Applicable

[:SENSe]:CORRection:CKIT <string>

(Read and Write) Set or query the calibration kit. The calibration kit can be “AV31101A”, “AV20201A”, and “AV20201B”. Wherein the suffix “A” represents the male head kit and the suffix “B” represents the female head kit. The query command returns the name of calibration kit.

Parameter Not Applicable

Example :CORR:CKIT AV31101A

Query Syntax :CORR:CKIT?

Default Not Applicable

Return Type Character string (char[])

Menu Operation 【Cal】－［M Kit］

[:SENSe]:FREQuency:CENTer

(Read and Write) Set or query the center frequency

Parameter Value of center frequency in Hz, the range of which for 3680A is: 1MHz~4GHz, and for 3680B is: 1MHz~8GHz.

Example :FREQ:CENT 50000000

Query Syntax :FREQ:CENT?

Default 2000 500 000 (2.0005GHz)

Return Type Numeric (double)

Menu Operation 【Freq/Dist】 — ［ Center Freq］

[:SENSe]:FREQuency:SPAN

(Read and Write) Set or query the span.

Parameter Value of span in Hz, the range of which for 3680A is: 0Hz~3.999GHz, and for 3680B is: 0Hz~7.999GHz.

Example :FREQ:SPAN 50000000

Query Syntax :FREQ:SPAN?

Default 3999 000 000 (3.999GHz)

Return Type Numeric (double)

Menu Operation 【 Freq/Dist】 — ［Span］

[:SENSe]:FREQuency:STARt

(Read and Write) Set or query the start frequency.

Parameter Value of start frequency in Hz ,the range of which for 3680A is: 1MHz~4GHz, and for 3680B is: 1MHz~8GHz.

Example :FREQ:STAR 50000000

Query Syntax :FREQ:STAR?

Default 1 000 000 (1MHz)

Return Type Numeric (double)

Menu Operation 【 Freq/Dist】 — ［Start Freq］

[:SENSe]:FREQuency:STOP

(Read and Write) Set or query the stop frequency.

Parameter Value of stop frequency in Hz, the range of which for 3680A is: 1MHz~4GHz, and for 3680B is: 1MHz~8GHz.

Example :FREQ:STOP 50000000

Query Syntax :FREQ:STOP?

Default 4000 000 000 (4GHz)

Return Type Numeric (double)

Menu Operation 【Freq/Dist】 — ［Stop Freq］

[:SENSe]:ROSCillator:SOURce <string>

(Read and Write) Set or query the frequency reference mode.

Parameter

Frequency reference mode	Set parameter	Return value

	<string>	<int>
Internal Ref Ref Output: Off	OFF	0
Internal Ref Ref Output: On	INTernal	1
External Ref	EXTernal	2

Example :ROSC:SOUR INTernal

Query Syntax :ROSC:SOUR?

Default OFF

Return Type Numeric (int)

Menu Operation 【System/Local】—［Next］—［Freq Ref Int Ext］，［Ref Out Off On］

[:SENSe]:SEGment:ADD

(Write Only) Add the list sweep segment.

Parameter Not Applicable

Example :SEG:ADD

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Sweep/Setup】—［Edit List］—［Add Seg］

[:SENSe]:SEGment:COUNt

(Read Only) Query the count of list sweep segment.

Parameter Not Applicable

Example :SEG:COUN?

Query Syntax :SEG:COUN?

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Sweep/Setup】—［List Info Off On］

[:SENSe]:SEGment[X]:DELete

(Write Only) Delete the segment X of list sweep.

Parameter Not Applicable

Example :SEG2:DEL

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Sweep/Setup】—［Edit List］—［Del Seg］

[:SENSe]:SEGment:DELete:ALL

(Write Only) Delete all the segments of list sweep.

Parameter Not Applicable

Example :SEG:DEL:ALL

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Sweep/Setup】－［Edit List］－［Clear］

[:SENSe]:SEGment[X]:FREQuency:STARt <num>

(Read and Write) Set or query the start frequency of segment X.

Parameter Value of start frequency in Hz, the range of which for 3680A is: 1MHz~4GHz, and for 3680B is: 1MHz~8GHz.

Example :SEG5:FREQ:STAR 50000000

Query Syntax :SEG5:FREQ:STAR?

Default 1 000 000 000 (1GHz)

Return Type Numeric (double)

Menu Operation 【Sweep/Setup】－［Edit List］－［Start Freq］

[:SENSe]:SEGment[X]:FREQuency:STOP <num>

(Read and Write) Set or query the stop frequency of segment X.

Parameter Value of stop frequency in Hz, the range of which for 3680A is: 1MHz~4GHz, and for 3680B is: 1MHz~8GHz.

Example :SEG5:FREQ:STOP 50000000

Query Syntax :SEG5:FREQ:STOP?

Default 2000 000 000 (2GHz)

Return Type Numeric (double)

Menu Operation 【Sweep/Setup】－［Edit List］－［Stop Freq］

[:SENSe]:SEGment[X]:SWEep:POINts <num>

(Read and Write) Set or query the sweep points of segment X.

Parameter Sweep points, and the sweep points of each segment should be less than 4001.

Example :SEG5:SWE:POIN 201

Query Syntax :SEG5:SWE:POIN?

Default 101

Return Type Numeric (int)

Menu Operation 【SWEEP/SET】－［Edit List］－［Sweep Points］

[:SENSe]:SEGment[X]:STATe <ON | OFF>

(Read and Write) Set or query the On or Off of segment X.

Parameter

ON or OFF of segment X	Set parameter <ON | OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example SEG5:STAT ON

Query Syntax SEG5:STAT?

Default OFF

Return Type Numeric (int)

Menu Operation Not Applicable

[:SENSe]:SEGment[X]:DATA

(Read Only) Query the sweep parameter of segment X. The format of returned data is “#NXXXX Data”, wherein XXXX represents the size of binary data. N represents the number of bits of XXXX. For example, # 3512 ... means the number of bits of binary data is 3, and the three-digit number following “3” is 512 which means that there is a binary data of 512 bytes thereafter.

It is defined as a structure below, and converts this binary data format into the data for this structure format.

```
typedef struct _SCAN_PARAM
{
        double      dStartFq;
        double      dStopFq;
        int         nPoint;
        BOOL        bSegOn;
}SCAN_PARAM;
```

Parameter Not Applicable

Example :SEG5:DATA?

Query Syntax :SEG5:DATA?

Default Not Applicable

Return Type Character array <char[]>

Menu Operation Not Applicable

[:SENSe]:SEGment:DATA:ALL

(Read Only) Query the sweep parameter of all segments. The format of returned data is “#NXXXX Data”, wherein XXXX represents the size of binary data. N represents the number of bits of XXXX. For example, # 3512 ... means the number of bits of binary data is 3, and the three-digit number following “3” is 512 which means that there is a binary data of 512 bytes thereafter.

It is defined as a structure below, and converts this binary data format into the data for this structure format.

```
typedef struct _SCAN_PARAM
{
        double     dStartFq;
        double     dStopFq;
        int        nPoint;
        BOOL       bSegOn;
}SCAN_PARAM;
```

Parameter Not Applicable

Example :SEG:DATA:ALL?

Query Syntax :SEG:DATA:ALL?

Default Not Applicable

Return Type Character array<char[]>

Menu Operation Not Applicable

[:SENSe]:SWEep:TYPE <string>

(Read and Write) Set or query the sweep type.

Parameter

Sweep type	Set parameter < sring >	Return value <int>
Linear sweep	LIN	0
List sweep	LIST	1

Example :SWE:TYPE LIN

Query Syntax :SWE:TYPE?

Default LIN

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［Swp Mode Linear List］

[:SENSe]:SWEep:POINts

(Read and Write) Set or query the points of linear sweep.

Parameter Sweep points, the range of which is 2~4001.

Example :SWE:POIN 500

Query Syntax :SWE:POIN?

Default 201

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［Points］

[:SENSe]:SWEep:TIME

(Read and Write) Set or query the time of linear sweep.

Parameter Sweep time, in ms.

Example :SWE:TIME 5000

Query Syntax :SWE:TIME?

Default 517

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［Swp Time Auto Man］

[:SENSe]:SWEep:TIME:AUTO <ON | OFF>

(Read and Write) Set or query the time mode of linear sweep.

Parameter

Sweep time mode	Set parameter < N | OFF>	Return value <int>
Manual	OFF	0
Auto	ON	1

Example :SWE:TIME:AUTO ON

Query Syntax :SWE:TIME:AUTO?

Default ON

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［Swp Time Auto Man］

2.2.5 :STATus Subsystem

:STATus:OPERation

(Read Only) Query whether the first sweep has ended.

Parameter Not Applicable

Example :STAT:OPER?

Query Syntax :STAT:OPER?

Default Not Applicable

Return Type Not Applicable

Menu Operation Not Applicable

2.2.6 :MMEMory Subsystem

:MMEMory:DELete:STATe <string>

(Write Only) Delete the state file in the current memory location.

Parameter File name without suffix.

Example :MMEM:DEL:STAT statefile

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Save/Recall】－［Recall State］－［Del］

:MMEMory:DELete:STATe:ALL

(Write Only) Delete all the state files in the current memory location.

Parameter	Not Applicable
Example	:MMEM:DEL:STAT:ALL
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Save/Recall】—［Recall State］—［Del All］

:MMEMory:DELete:LIMit <string>

(Write Only) Delete the limit line file in the current memory location.

Parameter	File name without suffix
Example	:MMEM:DEL:LIM lmtfile
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Limit】—［Recall］—［Del］

:MMEMory:DELete:LIMit:ALL

(Write Only) Delete all the limit line files in the current memory location.

Parameter	Not Applicable
Example	:MMEM:DEL:LIM:ALL
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Limit】—［Recall］—［Del All］

:MMEMory:DELete:TRACe <string>

(Write Only) Delete the trace file in the current memory location.

Parameter	File name without suffix
Example	:MMEM:DEL:TRAC tracefile
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Save/Recall】—［Recall Trace］—［Del］

:MMEMory:DELete:TRACe:ALL

(Write Only) Delete all the trace files in the current memory location.

Parameter	Not Applicable
Example	:MMEM:DEL:TRAC:ALL
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Save/Recall】－［Recall Trace］－［Del All］

:MMEMory:DELete:SCReen <string>

(Write Only) Delete the screen capture file in the current memory location.

Parameter	File name without suffix
Example	:MMEM:DEL:SRC jpegFile
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	Not Applicable

:MMEMory:DELete:SCReen:ALL

(Write Only) Delete all the screen capture files in the current memory location.

Parameter	Not Applicable
Example	:MMEM:DEL:SCReen:ALL
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	Not Applicable

:MMEMory:LOAD:LIMit <string>

(Write Only) Recall the limit line file in the current memory location.

Parameter	File name without suffix
Example	:MMEM:LOAD:LIM lmtfile
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Limit】－［Recall］－［Recall］

:MMEMory:LOAD:STATe <string>

(Write Only) Recall the state file in the current memory location.

Parameter	File name without suffix

Example :MMEM:LOAD:STAT statefile

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Save/Recall】－［Recall State］－［Recall］

:MMEMory:LOAD:TRACe <string>

(Write Only) Recall the trace file in the current memory location.

Parameter File name without suffix

Example :MMEM:LOAD:TRAC tracefile

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Save/Recall】－［Recall Trace］－［Recall］

:MMEMory:STORe:LIMit <string>

(Write Only) Store the limit line into the current memory location.

Parameter File name without suffix

Example :MMEM:STOR:LIM lmtfile

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Limit】－［］

:MMEMory:STORe:STATe <string>

(Write Only) Store the state into the current memory location.

Parameter File name without suffix

Example :MMEM:STOR:STAT statefile

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Save/Recall】－［Save State］

:MMEMory:STORe:TRACe <string>

(Write Only) Store trace into the current memory location.

Parameter File name without suffix

Example :MMEM:STOR:TRAC tracefile

Query Syntax Not Applicable

Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Save/Recall】－［Save Trace］

2.3 Specific Commands of Power Meter Mode

2.3.1 :CALCulate Subsystem

:CALCulate:RELative[:MAGNitude]

(Read Only) Query the relative measured value.

Parameter Not Applicable

Example :CALC:REL?

Query Syntax :CALC:REL?

Default Not Applicable

Return Type Numeric (double)

Menu Operation Not Applicable

:CALCulate:RELative:AUTO <ON | OFF>

(Read and Write) Set or query the ON or OFF of relative measurement.

Parameter

Relative measurement ON/OFF	Set parameter <ON | O F>	Return value <int>
Off	OFF	0
On	ON	1

Example :CALC:REL:AUTO ON

Query Syntax :CALC:REL:AUTO?

Default OFF

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［Relative. Off On］

2.3.2 :DISPlay Subsystem

:DISPlay:WINDow:ANALog:LOWer

(Read and Write) Set or query the minimum scale.

Parameter Minimum scale

Example :DISP:WIND:ANAL:LOW 5

Query Syntax :DISP:WIND:ANAL:LOW?

Default Not Applicable

Return Type Numeric (double)

Menu Operation 【Ampt】－［Min Scale］

:DISPlay:WINDow:ANALog:UPPer

(Read and Write) Set or query the maximum scale.

Parameter	Maximum scale
Example	:DISP:WIND:ANAL:UPP 5
Query Syntax	:DISP:WIND:ANAL:UPP?
Default	Not Applicable
Return Type	Numeric (double)
Menu Operation	【Ampt】－［Max Scale］

:DISPlay:WINDow:TRACe:Y:SCALe:AUTO

(Write Only) Set the amplitude range automatically.

Parameter	Not Applicable
Example	:DISP:WIND:TRAC:Y:SCAL:AUTO
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Ampt】－［Auto Scale］

2.3.3 [:SENSe] Subsystem

[:SENSe]:AVERage:COUNt <num>

(Read and Write) Set or query the average factor.

Parameter	Average factor, the range of which is (2~1000).
Example	:AVER:COUN 5
Query Syntax	:AVER:COUN?
Default	16
Return Type	Numeric (int)
Menu Operation	【Sweep/Setup】－［Avg Factor］

[:SENSe]:AVERage:STATe <ON | OFF>

(Read and Write) Set or query the ON or OFF of average.

Parameter Average On/Off.

Average On/Off	Set parameter <ON | OFF>	Return value <int>

Off	OFF	0
On	N	1

Example :AVER:STAT ON

Query Syntax :AVER:STAT?

Default OFF

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】 — ［Avg Off On］

[:SENSe]:AVERage:CLEar

(Write Only) Clear the average count (effective when average is ON).

Parameter Not Applicable

Example :AVER:CLE

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation Not Applicable

[:SENSe]:CORRection:GAIN <num>

(Read and Write) Set or query the offset value.

Parameter Offset value.

Example :CORR:GAIN 5

Query Syntax :CORR:GAIN?

Default 0

Return Type Numeric (double)

Menu Operation 【Ampt】 — ［Offset］

[:SENSe]:CORRection:GAIN:STATe <ON | OFF>

(Read and Write) Set or query On or Off of offset.

Parameter

Offset On/Off	Set parameter <ON | FF>	Return value <int>
Off	OFF	0
On	ON	1

Example :CORR:GAIN:STAT ON

Query Syntax :CORR:GAIN:STAT?

Default OFF

Return Type Numeric (int)

Menu Operation 【Sweep/Setup】－［Offset Off On］

[:SENSe]:FREQuency

(Read and Write) Set or query the frequency.

Parameter Frequency, in Hz.

Example :FREQ 50000000

Query Syntax :FREQ?

Default 10 000 000 (10MHz)

Return Type Numeric (double)

Menu Operation 【Freq/Dist】－［Frequency］

[:SENSe]:DATA

(Read Only) Query the measured value of power.

Parameter Not Applicable

Example :DATA?

Query Syntax :DATA?

Default Not Applicable

Return Type Numeric (double)

Menu Operation Not Applicable

2.3.4 :CALibration Subsystem

:CALibration:ZERO

(Write Only) Start zero calibration.

Parameter Not Applicable

Example :CAL:ZERO

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Cal】－［Zero］

:CALibration:ZERO:STATe

(Read Only) Query the state identification of zero calibration. 0: normal; 1: in process; 2: done; 3: failed.

Parameter Not Applicable

Example :CAL:ZERO:STAT?

Query Syntax :CAL:ZERO:STAT?

Default Not Applicable

Return Type Numeric (int)

Menu Operation Not Applicable

2.3.5 :MMEMory Subsystem

:MMEMory:DELete:STATe <string>

(Write Only) Delete the state file in the current memory location.

Parameter File name without suffix

Example :MMEM:DEL:STAT statefile

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Save/Recall】－［Recall State］－［Del］

:MMEMory:DELete:STATe:ALL

(Write Only) Delete all the state files in the current memory location.

Parameter Not Applicable

Example :MMEM:DEL:STAT:ALL

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Save/Recall】－［Recall State］－［Del All］

:MMEMory:STORe:STATe <string>

(Write Only) Store state in the current memory location.

Parameter File name without suffix

Example :MMEM:STOR:STAT statefile

Query Syntax Not Applicable

Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Save/Recall】－［Save State］

:MMEMory:LOAD:STATe <string>

(Write Only) Recall the state file in the current memory location.

Parameter	File name without suffix
Example	:MMEM:LOAD:STAT statefile
Query Syntax	Not Applicable
Default	Not Applicable
Return Type	Not Applicable
Menu Operation	【Save/Recall】－［Recall State］－［Recall］

2.4 Universal Commands

2.4.1 :INSTrument Subsystem

:INSTrument[:SELect] <string>

(Read and Write) Set or query the measurement mode.

Parameter

Measurement mode	Set parameter <string>	Return value <int>
Cable & Antenna Test	CAT	0
Power meter	USBPM	1

Example :INST CAT

Query Syntax :INST?

Default CAT

Return Type Numeric (int)

Menu Operation 【SYSTEM/Local】－［Meas Mode］

:INSTrument:CATalog

(Read Only) Query the available measurement mode. 0: No available mode; 1: on cable & antenna mode; 2: only power meter mode; 3: both modes are available.

Parameter Not Applicable

Example :INST:CAT?

Query Syntax :INST:CAT?

Default 1

Return Type Numeric (int)

Menu Operation 【System/Local】－［Meas Mode］

2.4.2 :FORMat Subsystem

:FORMat[:DATA] <string>

(Read and Write) The format of command returned data

The format of binary data is "#NXXXX Data", wherein XXXX represents the size of binary data. N represents the number of bits of XXXX. For example, # 3512 ... means the number of bits of binary data is 3, and the three-digit number following "3" is 512 which means that there is a binary data of 512 bytes thereafter. Then store such binary data into the variable defined by corresponding data type.

If the format is ASCII code, data is ended by a line break followed by one null character, i.e. "\n\0". If the received data is an array type, each element is separated by a "," (comma) in between.

Parameter

Data format	Set parameter <string>	Return value <int>
ASCII code	ASC	0
Binary	HEX	1

Example :FORM ASC

Query Syntax :FORM?

Default Binary (HEX)

Return Type Numeric (int)

Menu Operation Not Applicable

2.4.3 :MMEMory Subsystem

:MMEMory:STORe:SCREen <string>

(Write Only) Screen capture (jpg), stored at the current memory location.

Parameter File name without suffix

Example :MMEM:STOR:SCRE screenfile

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation 【Save/Recall】－［Print Screen］

:MMEMory:LOCation <string>

(Read and Write) Set or query memory location。

Parameter

Memory location	Set parameter <string>	Return value <int>
U disc	USB	0
SD card	SDCARD	1
Inside instrument	INT	2

Example :MMEM:LOC INT

Query Syntax :MMEM:LOC?

Default INT

Return Type Numeric (int)

Menu Operation 【Save/Recall】－［Location］

2.4.4 :SYSTem Subsystem

[:SYSTem]:GPS <ON | OFF>

(Read and Write) Query or set ON or OFF of GPS, if it is ON, the screen will display the latitude, longitude and altitude acquired by GPS. This command is an overlapped command, and it may use the command OPC? to query whether this command has been done before sending other commands.

Parameter

GPS On/Off	Set parameter <ON ｜OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :GPS ON

Query Syntax :GPS?

Default OFF

Return Type Numeric (int)

Menu Operation 【System/Local】－［Next］－［GPS］－［GPS Off On］

[:SYSTem]:GPS:DATA

(Read Only) Return the current GPS data in the following format: “<long.>,<lat>,<alt.>,<time UTC>”. For example: “38 28’11.22” N,122 42’13.23” W,152,06/28/2010 23:35:38\n”, it will return “--,--,--,--\n” if there is no data.

Parameter Not Applicable

Example :GPS:DATA?

Query Syntax :GPS:DATA?

Default Not Applicable

Return Type Character

Menu Operation 【System/Local】－［Next］－［GPS］－［Info］

[:SYSTem]:GPS:RECeive[:STATe]

(Read Only) Query the state of GPS receiver.

Parameter Not Applicable

Example :GPS:REC?

Query Syntax :GPS:REC?

Default Not Applicable

Return Type Numeric (int)

0: the receiver has no data.

1: the receiver has data.

Menu Operation Not Applicable

[:SYSTem]:GPS:RST

(Write Only) GPS cold start, which may accelerate the startup by new GPS if no GPS data is available for a long period in places with very poor signal.

Parameter Not Applicable

Example :GPS:RST

Query Syntax Not Applicable

Default Not Applicable

Return Type Not Applicable

Menu Operation Not Applicable

[:SYSTem]:GPS:STATe

(Read Only) Query the state of GPS.

Parameter Not Applicable

Example :GPS:STAT?

Query Syntax :GPS:STAT?

Default Not Applicable

Return Type Numeric (int)

0: no positioning

1: non differential positioning

2: differential positioning

3: ineffective PPS

4: estimate

Menu Operation Not Applicable

[:SYSTem]:PWR:SHUTdown<num>

(Read and Write) Query or set the shutdown time.

Parameter Value of shutdown time in minute, the range of which is [15,1200] minutes.

Example :PWR:SHUT 20

Query Syntax :PWR:SHUT?

Default 15

Return Type Numeric (int)

Menu Operation 【System/Local】－［Next］－［Power Manager］－［Shutdown Time］

[:SYSTem]:PWR:SHUTdown:STATe <ON | OFF>

(Read and Write) Query or set the ON or OFF of shutdown time, if it is ON, the instrument will shut down automatically when the set shutdown time is out.

Parameter

Automatic shutdown On/Off	Set parameter < N | OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :PWR:SHUT:STAT OFF

Query Syntax :PWR:SHUT:STAT?

Default OFF

Return Type Numeric (int)

Menu Operation 【System/Local】－［Next］－［Power Manager］－［Shutdown Time］

[:SYSTem]:PWR:SLEep <num>

(Read and Write) Query or set the sleep time.

Parameter Value of sleep time in minute, the range of which is [1,600] minutes.

Example :PWR:SLE 20

Query Syntax PWR:SLE?

Default 1

Return Type Numeric (int)

Menu Operation 【System/Local】－［Next］－［Power Manager］－［Sleep Time］

[:SYSTem]:PWR:SLEep:STATe <ON | OFF>

(Read and Write) Query or set the ON or OFF of sleep time, if it is ON, the instrument will sleep automatically with screen display turned off when the set sleep time is out (retime in case of sleep due to any operation).

Parameter

Sleep On or Off	Set parameter <ON | OFF>	Return value <int>
Off	OFF	0
On	ON	1

Example :PWR:SLE:STAT OFF

Query Syntax :PWR:SLE:STAT?

Default OFF

Return Type Numeric (int)

Menu Operation 【System/Local】－［Next］－［Power Manager］－［Sleep Time］

[:SYSTem]:TEMPerature

(Read Only) Query the instrument internal temperature.

Parameter Not Applicable

Example :TEMP?

Query Syntax :TEMP?

Default Not Applicable

Return Type Numeric (int)

Menu Operation 【System/Local】-[Self Test]

Chapter 3 Use of Secondary Development Library

For the convenience of users, we packaged the SCPI command to make it to dynamic link library, so that the users can set or query 3680A/B by recalling the dynamic link library conveniently, which is suitable for users to build automatic test system. In addition, we also provide 3680A/B tools software based on this dynamic library for users to perform remote control on the instrument.

Before using this secondary development library, it should first set up a development environment, including installation of the required library files and instrument drivers, etc., which will be introduced in section 3.1.

Sections 3.2 to 3.6 specifically introduce the functions contained in the secondary development library, including instrument connection commands, SCPI command commands, specific commands of cable & antenna measurement mode, specific commands of power measurement mode and universal commands of all the modes, etc. Introduction of the specific commands in two measurement modes is subdivided according to their superior menu for ease of reference. In addition, when using these functions, it should first obtain the instrument handle by the function "Connect the instrument" so as to provide the parameter required by other commands. Finally, it should close the instrument handle by the function "Disconnect the instrument" to end the programmed process.

3.1 Building Development Environment

This dynamic library is based on the VISA library of NI, and created under LabWindows/CVI 9.0, and the spanned files of dynamic library are AV3680A/B.h, AV3680A/B.dll and AV3680A/B.lib.

Configuration Requirements of Computer System:

- Inter Pentium 4 or higher processor
- Window2000/XP operating system
- 512Mbytes memory
- 500Mbytes hard disk space
- USB interface or LAN interface

Before the secondary development, it is required to build the development environment by the following steps:

1. Installation of tools software

This step will install VISA development library, CVI runtime library and files for programming environment on the computer. The installation file is the “AV3680A Tools.rar” in the accompanying CD.

2. Installation of USB driver

    a) Find out and right click on the file AV3680A USB.inf in the CD, and select “Install（I）” in the shortcut menu popped up;

    b) Connect the instrument, the system will pop up “Found New Hardware Wizard”, select “Install from a list or specific location(Advanced)” and click on the “Next”;

    c) Select “Don’t search.I will choose the driver to install”, and click on“Next”;

    d) Select model “AV3680A” from the list box, and click on “Next”;

    e) Wait until the installation of driver by wizard is finished, click on “Finish”.

Upon finishing above steps, the user may use the files of secondary development library, i.e. AV3680A.h, AV3680A.dll and AV3680A.lib. In addition, the user may also use the “3680ATools.exe” for installation to achieve the programmed control of instrument.

3.2 Instrument Connection Commands

Connect the instrument

Function Prototype *ViStatus _VI_FUNC AV3680A_Open (ViRsrc resourceName, ViSession* handle)*

Function Purpose Turn on the I/O communication of the instrument.

This is the first step of other communication with instrument, which completes the following initialization operation: turn on the handle of module based on the interface and logic address information specified by the parameter resourceName to set up data channel for instrument connection.

Note: Connect the instrument will set the format of return parameter to numeric format, please do not set it to the character format manually, otherwise error will occur when using the query function.

Input Parameter *resourceName*—Instrument resource character string.

If connection is made through USB cable, then the resource character string is: "USB0::0x1286::0xA6CD::?*::RAW". Wherein the wild card ?* is subject to the value displayed in device manager.

TCP resource character string is "TCPIP::172.141.11.202::5000::SOCKET", the underlined part is the default IP address of the instrument, if such address has been changed, the actual IP address shall prevail.

Output Parameter *handle*—Instrument communication handle.

Return value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Disconnect the instrument

Function Prototype *ViStatus _VI_FUNC AV3680A_Close (ViSession handle)*

Function Purpose Turn off the I/O communication of instrument.

It is necessary to recall this function to disconnect the instrument after the end of control over the instrument.

Input Parameter *handle*—Instrument communication handle.

Return value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.3 SCPI Common Commands

There are six SCPI common commands introduced in section 2.1, in this section only the query of instrument identification and restoring instrument default state are encapsulated.

Query instrument identification

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryIDN (ViSession handle, ViChar IDN[])*

Function Purpose Obtain the identification information of the instrument.

Including manufacturer, model, SN and application version, etc.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *IDN[]*—Instrument identification character string.

Format is: CETC41, AV3680A, SN, Application Version.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Reset instrument default state

Function Prototype *ViStatus _VI_FUNC AV3680A_Reset (ViSession handle)*

Function Purpose Set the instrument to the default state of current measurement mode.

Input Parameter *handle*—Instrument communication handle.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4 Specific Commands of Cable & Antenna Test

3.4.1 [Measure]

Set the measurement mode

Function Prototype *ViStatus _VI_FUNC AV3680A_SetMeasFormat(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the measurement mode.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Measurement mode, corresponding relation is:

0: Return Loss (log)

1: SWR

2: Cable Loss

3: DTF- SWR

4: DTF- Return Loss

5: Smith Chart

6: Phase Diagram

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the measurement mode

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryMeasFormat(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the measurement mode.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Measurement mode, corresponding relation is:

0: Return Loss (log)

1: SWR

2: Cable Loss

3: DTF- SWR

4: DTF- Return Loss

5: Smith Chart

6: Phase Diagram

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the number of window

Function Prototype *ViStatus _VI_FUNC AV3680A_SetDualWnd(ViSession handle, ViBoolean bVal)*

Function Purpose Set to single or dual window display, which can be used for observing two different measurement modes at the same time.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—Number of window, corresponding relation is:

0: Single

1: Dual

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the number of window

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryDualWnd(ViSession handle, ViBoolean bVal[])*

Function Purpose Query the number of window.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal []*—Number of window, corresponding relation is:

0: Single

1: Dual

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the activated window

Function Prototype *ViStatus _VI_FUNC AV3680A_SetCurWnd(ViSession handle, ViUInt32 nVal)*

Function Purpose It is used to activate top or bottom window for dual window display.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*— Window identification, corresponding relation is:

0: Top

1: Bottom

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the activated window

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCurWnd(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the current activated window.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Identification of current activated window, corresponding relation is:

0: Top

1: Bottom

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.2 [Sweep/Set]

Set the sweep type

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSwpType(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the sweep type to single or continuous.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Sweep type, corresponding relation is:

0: Single

1: Continuous

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the sweep type

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySwpType(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the current sweep type.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Sweep type, corresponding relation is:

0: Single

1: Continuous

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Immediately trigger

Function Prototype *ViStatus _VI_FUNC AV3680A_TrigerSwp(ViSession handle)*

Function Purpose Trigger sweep once again, if the mode is single, sweep will stop after end, if the mode is continuous, continuous sweep will start.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Run/Hold

Function Prototype *ViStatus _VI_FUNC AV3680A_HoldSwp(ViSession handle,*
ViBoolean bVal)

Function Purpose Set the current sweep to Hold or Run state, if it is Hold, sweep will stop without change in trace for convenience of observation.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—Sweep state, corresponding relation is:

0: Run

1: Hold

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the sweep state

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryHoldSwp(ViSession handle,*
ViBoolean bVal[])

Function Purpose Query the current sweep state.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Sweep state, corresponding relation is:

0: Run

1: Hold

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the sweep time

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSwpTime(ViSession handle,*
ViUInt32 nVal)

Function Purpose Set the sweep time under linear sweep.

If the current sweep time is set to Auto, recalling this command will switch it to

Manual mode.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*— Value of sweep time, in ms.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the sweep time

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySwpTime(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the sweep time under linear sweep.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Value of sweep time, in ms.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the setting mode of sweep time

Function Prototype *ViStatus _VI_FUNC AV3680A_SetAutoSwpTimeOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set the setting mode of linear sweep time to Auto or Manual.

In Auto mode, the instrument will automatically set to the minimum sweep time.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—Setting mode of sweep time, corresponding relation is:

0: Manual

1: Auto

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the setting mode of sweep time

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryAutoSwpTimeOn (ViSession handle, ViBoolean bVal[])*

Function Purpose Query the setting mode of linear sweep time

Input Parameter *handle*—Instrument communication handle.

OutPut Parameter *bVal[]*—Setting mode of sweep time, corresponding relation is:

0: Manual

1: Auto

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the sweep points

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSwpPoints (ViSession handle, ViUInt32 nVal)*

Function Purpose Set the sweep points under linear sweep. The range of sweep points is 2~4001.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Sweep points.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the sweep points

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySwpPoints (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the sweep points under linear sweep.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Sweep points.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the sweep mode

Function Prototype *ViStatus _VI_FUNC AV3680A_SetScanType(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the sweep mode to linear or list sweep.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Sweep mode, corresponding relation is:

0: Linear sweep

1: List sweep

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the sweep mode

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryScanType(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the sweep mode.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Sweep mode, corresponding relation is:

0: Linear sweep

1: List sweep

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Add a segment

Function Prototype *ViStatus _VI_FUNC AV3680A_AddSegment(ViSession handle)*

Function Purpose It is used for add a new segment in editing sweep list.

The default sweep parameter of such segment is: frequency 1G~2GHz, sweep points 101, but the total sweep points of all the segments cannot exceed 4001, and the program will automatically adjust the sweep points of the new segment based on this requirement.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete the specified segment

Function Prototype *ViStatus _VI_FUNC AV3680A_DelSegment(ViSession handle, ViUInt32 nSegId);*

Function Purpose It is used to delete the specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be set.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete all the segments

Function Prototype *ViStatus _VI_FUNC AV3680A_DelAllSegment(ViSession handle)*

Function Purpose It is used to delete all the sweep segments in editing sweep list.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the count of segments

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySegCount(ViSession handle,*
ViUInt32 nSegCount[])

Function Purpose Query the count of sweep segments in sweep list.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nSegCount[]*—Segment count.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the ON or OFF of segment

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSegState(ViSession handle,*
ViUInt32 nSegId,
ViBoolean bVal)

Function Purpose It is used to set the ON or OFF of the specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be set.

Input Parameter *bVal*—Sweep segment ON/OFF, corresponding relation is:

0: OFF

1: ON

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the ON or OFF of segment

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySegState(ViSession handle,*
ViUInt32 nSegId,
ViBoolean bVal[])

Function Purpose It is used to query the ON or OFF of specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be queried.

Output Parameter *bVal[]*—Sweep segment ON/OFF, corresponding relation is:

0: OFF

1: ON

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the start frequency of segment

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSegStartFq(ViSession handle,*
ViUInt32 nSegId,
ViReal64 fVal)

Function Purpose It is used to set the starting frequency of specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be set.

Input Parameter *fVal*—Value of starting frequency, in Hz.
For 3680A, the range of which is 1MHz~4GHz.
For 3680B, the range of which is 1MHz~8GHz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the start frequency of segment

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySegStartFq(ViSession handle,*
ViUInt32 nSegId,
ViReal64 fVal[])

Function Purpose It is used to query the start frequency of specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be queried.

Output Parameter *fVal[]*—Value of start frequency, in Hz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the stop frequency of segment

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSegStopFq(ViSession handle,*
ViUInt32 nSegId,
ViReal64 fVal)

Function Purpose It is used to set the stop frequency of specified segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be set.

Input Parameter *fVal*— Value of stop frequency, in Hz.
For 3680A, the range of which is 1MHz~4GHz.
For 3680B, the range of which is 1MHz~8GHz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the stop frequency of segment

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySegStopFq(ViSession handle,*
ViUInt32 nSegId,
ViReal64 fVal[])

Function Purpose It is used to query the stop frequency of specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*— Sweep segment to be queried.

Output Parameter *fVal[]*—Value of stop frequency, in Hz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the segment sweep points

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSegPoints(ViSession handle,*
ViUInt32 nSegId,
ViUInt32 nVal)

Function Purpose It is used to set the sweep points of specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be set.

Input Parameter *nVal*— Sweep points. The set value will be adjusted based on that the total sweep points cannot exceed 4001.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the segment sweep points

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySegPoints(ViSession handle,*
ViUInt32 nSegId,
ViUInt32 nVal[])

Function Purpose It is used to query the sweep points of specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be queried.

Output Parameter *nVal []*—Sweep points.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the parameter of specified segment

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySegParam(ViSession handle,*
ViUInt32 nSegId,
SCAN_PARAM param[])

Function Purpose It is used to query the sweep parameter of specified sweep segment in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nSegId*—Sweep segment to be queried.

Output Parameter *param[]*—Sweep parameter. This parameter is the SCAN_PARAM self-defined structure type, defined as follows:

```
typedef struct _SCAN_PARAM
{
      double    dStartFq;
      double    dStopFq;
      int       nPoint;
      BOOL      bSegOn;
}SCAN_PARAM;
```

Sweep parameters contained are: start frequency, stop frequency, sweep points and segment ON/OFF.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the parameter of all segments

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryAllSegParams(ViSession handle,*
SCAN_PARAM params[],
ViUInt32 nSegCount[])

Function Purpose It is used to query the sweep parameter of all the sweep segments in editing sweep list.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *params[]*—Sweep parameter array. This array is SCAN_PARAM self-defined structure type, defined as follows:

```
typedef struct _SCAN_PARAM
{
    double    dStartFq;
    double    dStopFq;
    int       nPoint;
    BOOL      bSegOn;
}SCAN_PARAM;
```

Sweep parameters contained are: start frequency, stop frequency, sweep points and segment ON/OFF.

The count of array is obtained by the parameter *nSegCount*.

Output Parameter *nSegCount[]*－Total count of sweep segments, i.e. the number of elements in parameter *params[]* array.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On/Off of average

Function Prototype *ViStatus _VI_FUNC AV3680A_SetAvgOn (ViSession handle, ViBoolean bVal)*

Function Purpose Set the On/Off of averaging.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—Average On/Off, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On/Off of average

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryAvgOn(ViSession handle, ViBoolean bVal[])*

Function Purpose Query the On/Off of averaging.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Average On/Off, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the average factor

Function Prototype *ViStatus _VI_FUNC AV3680A_SetAvgFactor (ViSession handle, ViUInt32 nVal)*

Function Purpose Set the average factor to be used in averaging.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Average factor, the range of which is 1~1000.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the average factor

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryAvgFactor (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the average factor

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Average factor.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Clear the average count

Function Prototype *ViStatus _VI_FUNC AV3680A_ClearAvgCount (ViSession handle)*

Function Purpose Clear the average count. If average is ON, averaging will restart after clear.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On/Off of smooth

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSmoothOn (ViSession handle, ViBoolean bVal)*

Function Purpose Set the ON/OFF of smoothing.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—Smooth On/Off, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On/Off of smooth

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySmoothOn(ViSession handle, ViBoolean bVal[])*

Function Purpose Query the On/Off of smoothing.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Smooth On/Off, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the smooth aperture

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSmoothAper(ViSession handle, ViReal64 fVal)*

Function Purpose Set the smooth aperture to be used in smoothing.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*— Percentage of smooth aperture, the range of which is 0.01~20.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the smooth aperture

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySmoothAper (ViSession handle,ViReal64 fVal[])*

Function Purpose Query the value of smooth aperature.

Input Parameter *handle*—Instrument communication handle.

OutputParameter *fVa[]*—Value of smooth aperture (%).

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the intermediate frequency bandwidth

Function Prototype *ViStatus _VI_FUNC AV3680A_SetIFBW(ViSession handle, ViReal64 fVal)*

Function Purpose Set the intermediate frequency bandwidth.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of intermediate frequency bandwidth, in Hz.

Effective IF bandwidth is a discrete value between 1Hz~10kHz, in step of 1-2-5. If the set value is not among those effective values, then the effective value that is closest to the set value shall be set as the IF bandwidth.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the intermediate frequency bandwidth

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryIFBW (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the intermediate frequency bandwidth.

Input Parameter *handle*—Instrument communication handle.

OutputParameter *fVa[]*—Value of intermediate frequency bandwidth, in Hz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.3 [Trace]

Set the trace mathematics (display) mode

Function Prototype *ViStatus _VI_FUNC AV3680A_SetTraceMathFunc(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the trace mathematics or display mode.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Trace mathematics or display type, corresponding relation is:

0: Data

1: Memory

2: Data & Memory

3: Data－Memory

4: Data/Memory

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the trace mathematics mode

Function Prototype *ViStatus_VI_FUNC AV3680A_QueryTraceMathFunc (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the trace mathematics or display mode.

Input Parameter *handle*—Instrument communication handle.

OutputParameter *nVal[]*—Trace mathematics or display type, corresponding relation is:

0: Data

1: Memory

2: Data & Memory

3: Data－Memory

4: Data/Memory

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Copy the current data as the memory trace

Function Prototype *ViStatus _VI_FUNC AV3680A_TraceToMemory (ViSession handle)*

Function Purpose Copy the current data as the memory trace

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.4 [Amplitude]

Set the top amplitude

Function Prototype *ViStatus _VI_FUNC AV3680A_SetScaleTop(ViSession handle, ViReal64 fVal)*

Function Purpose Set the top amplitude.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Top scale value.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the top amplitude

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryScaleTop (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the top amplitude.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Top scale value.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the bottom amplitude

Function Prototype *ViStatus _VI_FUNC AV3680A_SetScaleBottom(ViSession handle, ViReal64 fVal)*

Function Purpose Set the bottom amplitude.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Bottom scale value.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the bottom amplitude

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryScaleBottom (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the bottom amplitude

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Bottom scale value.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Adjust the amplitude range automatically

Function Prototype *ViStatus _VI_FUNC AV3680A_AutoScale(ViSession handle)*

Function Purpose Adjust the amplitude range automatically.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set to the default amplitude range

Function Prototype *ViStatus _VI_FUNC AV3680A_DefaultScale(ViSession handle)*

Function Purpose Set to the default amplitude range.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.5 [Frequency]

Set the start frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_SetStartFreq(ViSession handle, ViReal64 fVal)*

Function Purpose Set the start frequency of linear sweep.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of start frequency, in Hz.
For 3680A, the range of which is 1MHz~4GHz.
For 3680B, the range of which is 1MHz~8GHz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the start frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryStartFreq (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the start frequency.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of start frequency, in Hz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the stop frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_SetStopFreq(ViSession handle, ViReal64 fVal)*

Function Purpose Set the stop frequency of linear sweep.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of stop frequency, in Hz.
For 3680A, the range of which is 1MHz~4GHz.
For 3680B, the range of which is 1MHz~8GHz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the stop frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryStopFreq (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the stop frequency of linear sweep.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of stop frequency, in Hz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the center frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_SetCenterFreq(ViSession handle, ViReal64 fVal)*

Function Purpose Set the center frequency of linear sweep.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of center frequency, in Hz.
For 3680A, the range of which is 1MHz~4GHz.
For 3680B, the range of which is 1MHz~8GHz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the center frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCenterFreq (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the center frequency of linear sweep.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of center frequency, in Hz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the span

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSpan(ViSession handle, ViReal64 fVal)*

Function Purpose Set the span of linear sweep.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of span, in Hz.
For 3680A, the range of which is 0Hz~3.999GHz.

For 3680B, the range of which is 0Hz~7.999GHz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the span

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySpan (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the span of linear sweep.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of span, in Hz.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.6 [Distance]

Set the start distance

Function Prototype *ViStatus _VI_FUNC AV3680A_SetStartDist (ViSession handle, ViReal64 fVal)*

Function Purpose Set the start distance in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of start distance, in m or ft, depending on DTF setting.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the start distance

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryStartDist (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the start distance in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of start distance, in m or ft, depending on DTF setting.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the stop distance

Function Prototype *ViStatus _VI_FUNC AV3680A_SetStopDist (ViSession handle, ViReal64 fVal)*

Function Purpose Set the stop distance in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of stop distance, in m or ft, depending on DTF setting.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the stop distance

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryStopDist (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the stop distance in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of stop distance, in m or ft, depending on DTF setting.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the cable loss

Function Prototype *ViStatus _VI_FUNC AV3680A_SetCableLoss(ViSession handle, ViReal64 fVal)*

Function Purpose Set the cable loss in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of cable loss, in dB/m or dB/ft, depending on DTF setting. The range of which is 0~30dB/m.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the cable loss

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCableLoss (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the cable loss in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of cable loss, in dB/m or dB/ft, depending on DTF setting.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the velocity factor

Function Prototype *ViStatus _VI_FUNC AV3680A_SetVFactor (ViSession handle, ViReal64 fVal)*

Function Purpose Set the velocity factor in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of velocity factor, the range of which is 0.001~1.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the velocity factor

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryVFactor (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the velocity factor in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal*—Value of velocity factor.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the cable model

Function Prototype ViStatus _VI_FUNC AV3680A_SetCableMode (ViSession handle, ViUInt32 nVal)

Function Purpose Set the cable model via index.

Input Parameter *handle*—Instrument communication handle.

Input Parameter nVal—Cable model index.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the cable model

Function Prototype ViStatus _VI_FUNC AV3680A_QueryCableMode (ViSession handle, ViChar *pVal)

Function Purpose Query the cable model.

Input Parameter *handle*—Instrument communication handle.

Output Parameter pVal—Character string pointer directing cable model.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the unit of DTF

Function Prototype *ViStatus _VI_FUNC AV3680A_SetDTFUnit(ViSession handle,* *ViUInt32 nVal)*

Function Purpose Set the distance unit in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Distance unit, corresponding relation is:

0: m (meter)

1: ft (foot)

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the unit of DTF

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySetDTFUnit (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the distance unit in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Distance unit, corresponding relation is:

0: m (meter)

1: ft (foot)

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the window function

Function Prototype *ViStatus _VI_FUNC AV3680A_SetWinFunc(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the window function used in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Window function, corresponding relation is:

0: Rectangular window

1: Hanning window (normal edge smoothing)

2: Hamming window (low edge smoothing)

3: Blackman window (minimum edge smoothing)

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the window function

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryWinFunc (ViSession handle,*

ViUInt32 nVal[])

Function Purpose Query the window function used in DTF measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Window function, corresponding relation is:

0: Rectangular window

1: Hanning window (normal edge smoothing)

2: Hamming window (low edge smoothing)

3: Blackman window (minimum edge smoothing)

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.7 [Calibrate]

Set the On/Off of error correction

Function Prototype *ViStatus _VI_FUNC AV3680A_SetCalOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set the On/Off of error correction.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—On/Off of error correction, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On/Off of error correction

Function Prototype *ViStatus _VI_FUNC AV3680A_SetCalOn(ViSession handle, ViBoolean bVal)*

Function Purpose Query the On/Off of error correction.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—On/Off of error correction, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Acquire the "Open" measurement data

Function Prototype *ViStatus _VI_FUNC AV3680A_CalCollOpen (ViSession handle)*

Function Purpose Acquire the "Open" measurement data during calibration.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query whether the "Open" measurement data has been acquired

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCalCollOpen (ViSession handle, ViBoolean bVal[])*

Function Purpose Query whether the "Open" measurement data has been acquired during calibration.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Acquisition state, corresponding relation is:

0: Undone

1: Done

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Acquire the "Short" measurement data

Function Prototype *ViStatus _VI_FUNC AV3680A_CalCollShort (ViSession handle)*

Function Purpose Acquire the "Short" measurement data during calibration.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query whether the "Short" measurement data has been acquired

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCalCollShort (ViSession handle, ViBoolean bVal[])*

Function Purpose Query whether the "Short" measurement data has been acquired during calibration.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Acquisition state, corresponding relation is:

0: Undone

1: Done

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Acquire the "Load" measurement data

Function Prototype *ViStatus _VI_FUNC AV3680A_CalCollLoad (ViSession handle)*

Function Purpose Acquire the "Load" measurement data during calibration.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query whether the "Load" measurement data has been acquired

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCalCollLoad (ViSession handle, ViBoolean bVal[])*

Function Purpose Query whether the "Load" measurement data has been acquired during calibration.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Acquisition state, corresponding relation is:

0: Undone

1: Done

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Finish the data acquisition for calibration standard

Function Prototype *ViStatus _VI_FUNC AV3680A_CalCollFinish(ViSession handle)*

Function Purpose Calculate the error coefficient by executing this function after finishing the measurement data acquisition for three calibration standard during calibration.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Abort the data acquisition for calibration standard

Function Prototype *ViStatus _VI_FUNC AV3680A_CalCollAbort(ViSession handle)*

Function Purpose Abort the calibration process and release the resources occupied in calibration process. The calibration process will restore automatically when acquiring data of some calibration standard.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the validity of error coefficient

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCalValid (ViSession handle, ViBoolean bVal[])*

Function Purpose Query the validity of error coefficient. When the data of three calibration standard have been acquired and the command "Finish data acquisition of calibration standard" is executed for calculation of error coefficient, the error coefficient is valid.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Validity of error coefficient, corresponding relation is:

0: Invalid

1: Valid

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Select the mechanical calibration kit

Function Prototype *ViStatus _VI_FUNC AV3680A_SetCalKit (ViSession handle, ViUInt32 nVal)*

Function Purpose Select the model of mechanical calibration kit used, the suffix "A" represents male calibration kit, and "B" represents female calibration kit.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Model of calibration kit, corresponding relation is:

0: AV31101A

1: AV20101A

2: AV20101B

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the name of calibration kit

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCalKit (ViSession handle, ViChar *pVal)*

Function Purpose Query the name of calibration kit, the suffix "A" represents male calibration kit, and "B" represents female calibration kit.

Input Parameter *handle*—Instrument communication handle.

Output Parameter pVal—Pointer directing the character string of calibration kit name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.8 [Limit]

Set the limit line mode

Function Prototype *ViStatus _VI_FUNC AV3680A_SetLmtMode(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the mode of limit line to upper limit or lower limit.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Limit line mode, corresponding relation is:

1: Upper limit

0: Lower limit

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the limit line mode

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryLmtMode (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the test mode of limit line.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Limit line mode, corresponding relation is:

1: Upper limit

0: Lower limit

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On or Off of limit test

Function Prototype *ViStatus _VI_FUNC AV3680A_SeLmtTestOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set the On or Off of limit test.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—On or Off of limit test, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On or Off of limit test

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryLmtTestOn (ViSession handle,* *ViBoolean bVal[])*

Function Purpose Query the On or Off of limit test.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—On or Off of limit test, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the limit test result

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryLmtFail (ViSession handle, ViBoolean bVal[])*

Function Purpose Query the limit test result: Pass or Fail.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Limit test result, corresponding relation is:

0: Pass

1: Fail

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On or Off of limit alarm

Function Prototype *ViStatus _VI_FUNC AV3680A_SetAlarmOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set the On or Off of limit alarm.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—On or Off of limit alarm, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On or Off of limit alarm

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryAlarmOn (ViSession handle,* *ViBoolean bVal[])*

Function Purpose Query the On or Off of limit alarm.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—On or Off of limit alarm, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the number of limit points

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryLmtPtNum (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the number of limit points.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Number of limit points

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Add a limit point

Function Prototype *ViStatus _VI_FUNC AV3680A_LmtAddPt(ViSession handle)*

Function Purpose It is used to add a new limit point between the current limit point and the next limit point in editing limit line.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Clear all the limit points

Function Prototype *ViStatus _VI_FUNC AV3680A_LmtClear(ViSession handle)*

Function Purpose It is used to delete all the limit points in editing limit line.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete the current limit point

Function Prototype *ViStatus _VI_FUNC AV3680A_LmtDelPt(ViSession handle)*

Function Purpose It is used to delete the current limit point in editing limit line.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Select the limit point

Function Prototype *ViStatus _VI_FUNC AV3680A_LmtSelectPt(ViSession handle, ViUInt32 nVal)*

Function Purpose It is used to set the current limit point via serial number in editing limit line.

Input Parameter *nVal* —Serial number of limit point, the most left point is No.1, increased to the right.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the current limit point index

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryLmtSelectPt (ViSession handle, ViUInt32 nVal[])*

Function Purpose It is used to query the current limit point index in editing limit line.

OutPut Parameter *nVal[]* —Serial number of limit point, the most left point is No.1, increased to the right.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the X value of current limit point

Function Prototype *ViStatus _VI_FUNC AV3680A_SetLmtPtX(ViSession handle, ViReal64 fVal)*

Function Purpose Set the X-coordinate of current limit point (frequency or distance value).

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—X-coordinate value of limit point.

In frequency domain measurement (non DTF measurement), this value is the frequency value in Hz.

In time domain measurement (DTF measurement), this value is the distance value in unit (m or ft) set by current DTF.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the X value of current limit point

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryLmtPtX (ViSession handle,* *ViReal64 fVal[])*

Function Purpose Query the X-coordinate of current limit point (frequency or distance value).

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—X-coordinate value of limit point.

In frequency domain measurement (non DTF measurement), this value is the frequency value in Hz.

In time domain measurement (DTF measurement), this value is the distance value in unit (m or ft) set by current DTF.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the Y value of current limit point

Function Prototype *ViStatus _VI_FUNC AV3680A_SetLmtPtY(ViSession handle, ViReal64 fVal)*

Function Purpose Set the Y-coordinate of current limit point (amplitude).

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Y-coordinate value of limit point, in the same unit as the Y-coordinate unit in current measurement mode.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the Y value of current limit point

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryLmtPtY (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the Y-coordinate of current limit point (amplitude).

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Y-coordinate value of limit point, in the same unit as the Y-coordinate unit in current measurement mode.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.9 [Store/Recall]

Store state

Function Prototype	*ViStatus _VI_FUNC AV3680A_StoreStateFile(ViSession handle,* *ViChar* pVal)*
Function Purpose	Store the current parameter state as the state file with specified name.
Input Parameter	*handle*—Instrument communication handle.
Input Parameter	*pVal*—File name.
Return Value	Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Store trace

Function Prototype	*ViStatus _VI_FUNC AV3680A_StoreTraceFile(ViSession handle,* *ViChar* pVal)*
Function Purpose	Store the current trace as the trace file with specified name.
Input Parameter	*handle*—Instrument communication handle.
Input Parameter	*pVal*—File name.
Return Value	Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Store limit line

Function Prototype	*ViStatus _VI_FUNC AV3680A_StoreLmtFile(ViSession handle,* *ViChar* pVal)*
Function Purpose	Store the current limit line as the limit line file with specified name.
Input Parameter	*handle*—Instrument communication handle.
Input Parameter	*pVal*—File name.
Return Value	Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Recall a state file

Function Prototype	*ViStatus _VI_FUNC AV3680A_LoadStateFile(ViSession handle,* *ViChar* pVal)*
Function Purpose	Recall a state file.
Input Parameter	*handle*—Instrument communication handle.
Input Parameter	*pVal*—File name.
Return Value	Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Recall a trace file

Function Prototype	*ViStatus _VI_FUNC AV3680A_LoadTraceFile(ViSession handle,*

ViChar   pVal)*

Function Purpose Recall a trace file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Recall a limit line file

Function Prototype *ViStatus _VI_FUNC AV3680A_LoadLmtFile(ViSession handle,*
ViChar   pVal)*

Function Purpose Recall a limit file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete a single state file

Function Prototype *ViStatus _VI_FUNC AV3680A_DelStateFile(ViSession handle,*
ViChar   pVal)*

Function Purpose Delete a single state file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete a single trace file

Function Prototype *ViStatus _VI_FUNC AV3680A_DelTraceFile(ViSession handle,*
ViChar   pVal)*

Function Purpose Delete a single trace file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete a single limit line file

Function Prototype *ViStatus _VI_FUNC AV3680A_DelLmtFile(ViSession handle, ViChar* pVal)*

Function Purpose Delete a single limit line file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete all the state files

Function Prototype *ViStatus _VI_FUNC AV3680A_DelAllStateFile(ViSession handle)*

Function Purpose Delete all the state files.

Input Parameter *handle*—Instrument communication handle.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete all the trace files

Function Prototype *ViStatus _VI_FUNC AV3680A_DelAllTraceFile(ViSession handle)*

Function Purpose Delete all the trace files.

Input Parameter *handle*—Instrument communication handle.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete all the limit line files

Function Prototype *ViStatus _VI_FUNC AV3680A_DelAllLmtFile(ViSession handle)*

Function Purpose Delete all the limit line files.

Input Parameter *handle*—Instrument communication handle.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.10 [Marker]

Set the marker state

Function Prototype *ViStatus _VI_FUNC AV3680A_SetMkrState(ViSession handle,* *ViUInt32 nMkrId,ViUInt32 nMode)*

Function Purpose Set the state of specified marker to: Off, Normal, Delta.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nMkrId* —Marker identification, the value of which is an integral number between 1 and 6.

Input Parameter *nMode*—Marker mode, corresponding relation is:

0: Off

1: Normal

2: Delta

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the marker state

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryMkrState(ViSession handle,*
ViUInt32 nMkrId,
ViUInt32 nMode[])

Function Purpose Query the state of specified marker to: Off, Normal, Delta.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nMkrId* —Marker identification, the value of which is an integral number between 1 and 6,

Output Parameter *nMode[]*—Marker mode, corresponding relation is:

0: Off

1: Normal

2: Delta

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the marker to the maximum location

Function Prototype *ViStatus _VI_FUNC AV3680A_SetMkrMaxSearch (ViSession handle,*
ViUInt32 nMkrId)

Function Purpose Set the specified marker to the maximum location of trace. If this group of marker is not turned on, then turn on the reference marker and set it to the maximum location. If only the reference marker of this group is turned on, then set the reference marker to the maximum location. If delta marker is turned on, then set the delta marker to the maximum location.

Input Parameter *nMkrId* —Marker identification, the value of which is an integral number between 1 and 6.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the marker to the minimum location

Function Prototype *ViStatus _VI_FUNC AV3680A_SetMkrMinSearch(ViSession handle,*
ViUInt32 nMkrId)

Function Purpose Set the specified marker to the minimum location of trace. If this group of marker is not turned on, then turn on the reference marker and set it to the minimum location. If only the reference marker of this group is turned on, then set the reference marker to the minimum location. If delta marker is turned on, then set the delta marker to the minimum location.

Input Parameter *nMkrId* —Marker identification, the value of which is an integral number between 1 and 6.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Turn off all the markers

Function Prototype *ViStatus _VI_FUNC AV3680A_SetMkrAOff(ViSession handle)*

Function Purpose Turn off all the markers.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the X-coordinate value

Function Prototype *ViStatus _VI_FUNC AV3680A_SetMkrXVal(ViSession handle,*
ViUInt32 nMkrId,
ViReal64 fVal)

Function Purpose Set the X-coordinate value of specified marker.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nMkrId* —Marker identification, the value of which is an integral number between 1 and 6.

Input Parameter *fVal*—X-coordinate value of marker

In frequency domain measurement (non DTF measurement), this value is the frequency value in Hz.

In time domain measurement (DTF measurement), this value is the distance value in unit (m or ft) set by current DTF.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the X-coordinate value

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryMkrXVal(ViSession handle,*
ViUInt32 nMkrId,
ViReal64 fVal[])

Function Purpose Query the X-coordinate value of specified marker.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nMkrId* —Marker identification, the value of which is an integral number between 1 and 6.

Output Parameter *fVal[]*—X-coordinate value of marker

In frequency domain measurement (non DTF measurement), this value is the frequency value in Hz.

In time domain measurement (DTF measurement), this value is the distance value in unit (m or ft) set by current DTF.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the Y-coordinate value

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryMkrYVal(ViSession handle,*
ViUInt32 nMkrId,
float fVal[])

Function Purpose Query the Y-coordinate value of specified marker (amplitude)。

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nMkrId* —Marker identification, the value of which is an integral number between 1 and 6.

Output Parameter *fVal[]*—Y-coordinate value of marker, in the same unit as the Y-coordinate unit in current measurement mode.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.11 [System]

Set the frequency reference

Function Prototype *ViStatus _VI_FUNC AV3680A_SetRoscSourc (ViSession handle, ViUInt32 nVal)*

Function Purpose Set the frequency reference working mode.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Frequency reference working mode, corresponding relation is:

0: Internal reference, without output

1: Internal reference, with output

2: External reference, with input

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the frequency reference

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryRoscSourc (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the frequency reference working mode.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Frequency reference working mode, corresponding relation is:

0: Internal reference, without output

1: Internal reference, with output

2: External reference, with input

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.4.12 Others

Query whether the first sweep is finished

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySwpOneFinish (ViSession handle, ViBoolean bVal[])*

Function Purpose Query whether the first sweep is finished

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—Whether the first sweep is finished, corresponding relation is:

0: Not finished

1: Finished

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Read the current trace original data

Function Prototype *ViStatus _VI_FUNC AV3680A_ReadCurTrace(ViSession handle,*
float pData[],
ViInt32 nSize[])

Function Purpose Obtain the original data of current trace, without format conversion, averaging and smoothing. If the error correction is active, then such data is that gone through error correction.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *pData[]* —Array composed of data points.

The data point is a complex number, every two elements of the data compose the real part and imaginary part of a data point, and the forepart is the real part and the afterpart is the imaginary part.

Output Parameter *nSize[]*—pData*[]*number of elements in array, which shall be double sweep points.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Read the memory trace original data

Function Prototype *ViStatus _VI_FUNC AV3680A_ReadMemTrace(ViSession handle,*
float pData[],
ViInt32 nSize[])

Function Purpose Obtain the original data of memory trace, without format conversion, averaging and smoothing. If the error correction is active when storing memory trace, then such data is that gone through error correction.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *pData[]* —Array composed of data points

The data point is a complex number, every two elements of the data compose the real part and imaginary part of a data point, and the forepart is the real part and the afterpart is the imaginary part.

Output Parameter *nSize[]*—pData*[]*number of elements in array, which shall be double sweep points.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Read the current trace data

Function Prototype *ViStatus _VI_FUNC AV3680A_ReadCurTrace_Processed(*
ViSession handle,
float pData[],
ViInt32 nSize[])

Function Purpose Read the current trace data, with format conversion, averaging and smoothing.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *pData[]* —Array composed of data points

In the measurement format of Smith chart, the data point is a complex number, and every two elements in the array *pData[]* compose the real part and imaginary part of a data point, and the forepart is the real part and the afterpart is the imaginary part.

In the measurement format of non Smith chart, the trace data point is a real number and every element in the array *pData[]* corresponds to a data point.

Output Parameter *nSize[]*—pData*[]*number of elements in array.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Read the memory trace data

Function Prototype *ViStatus _VI_FUNC AV3680A_ReadMemTrace_Processed(*

ViSession handle,

float pData[],

ViInt32 nSize[])

Function Purpose Read the memory trace data. Such data the result after format conversion, averaging and smoothing of original data.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *pData[]* —Array composed of data points

In the measurement format of Smith chart, the data point is a complex number, and every two elements in the array *pData[]* compose the real part and imaginary part of a data point, and the forepart is the real part and the afterpart is the imaginary part.

In the measurement format of non Smith chart, the trace data point is a real number and every element in the array *pData[]* corresponds to a data point.

Output Parameter *nSize[]*—pData*[]*number of elements in array.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.5 Specific Commands of Power Meter

3.5.1 [Frequency]

Set the frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_SetFreq(ViSession handle, ViReal64 fVal)*

Function Purpose Set the frequency value of power meter measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Value of frequency, in Hz.
The range of which depends on the model of power sensor.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the frequency

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryFreq(ViSession handle, ViReal64 fVal[])*

Function Purpose Query the frequency value of power meter measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Value of frequency, in Hz.
The range of which depends on the model of power sensor.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.5.2 [Sweep/Set]

Set the offset value

Function Prototype *ViStatus _VI_FUNC AV3680A_SetOffset(ViSession handle, ViReal64 fVal)*

Function Purpose Set the offset value of power meter measurement.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *fVal*—Offset value.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the offset value

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryOffset(ViSession handle, ViReal64 fVal[])*

Function Purpose Query the offset value of power meter measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*— Offset value.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On or Off of offset

Function Prototype *ViStatus _VI_FUNC AV3680A_SetOffsetOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set the On or Off of offset.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—On or Off of offset, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On or Off of offset

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryOffsetOn(ViSession handle, ViBoolean bVal[])*

Function Purpose Query the On or Off of offset.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—On or Off of offset, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the reference value

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryRefVal(ViSession handle, ViReal64 fVal[])*

Function Purpose Query the reference value of power meter measurement.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Reference value

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On or Off of reference

Function Prototype *ViStatus _VI_FUNC AV3680A_SetRefOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set the On or Off of reference.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—On or Off of reference, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On or Off of reference

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryRefOn(ViSession handle, ViBoolean bVal[])*

Function Purpose Query the On or Off of reference.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—On or Off of reference, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.5.3 [Calibrate]

Zero calibration operation

Function Prototype *ViStatus _VI_FUNC AV3680A_SetCalibZero(ViSession handle)*

Function Purpose Start the zero calibration.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the zero calibration state

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryCalibZero (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the zero calibration state.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal []*—Zero calibration state, corresponding relation is:

0: Normal (no zero calibration)

1: In zero calibration

2: Done (will become normal zero calibration state after 5 seconds)

3: Fail (will become normal zero calibration state after 5 seconds)

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.5.4 [Store/Recall]

Store a state file

Function Prototype *ViStatus _VI_FUNC AV3680A_StoreStateFile(ViSession handle, ViChar* pVal)*

Function Purpose Store the current parameter state as the state file with specified name.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Recall a state file.

Function Prototype *ViStatus _VI_FUNC AV3680A_LoadStateFile(ViSession handle, ViChar* pVal)*

Function Purpose Recall a state file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete a single state file.

Function Prototype *ViStatus _VI_FUNC AV3680A_DelStateFile(ViSession handle, ViChar* pVal)*

Function Purpose Delete a single state file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete all the state files

Function Prototype *ViStatus _VI_FUNC AV3680A_DelAllStateFile(ViSession handle)*

Function Purpose Delete all the state files.

Input Parameter *handle*—Instrument communication handle.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.5.5 Others

Query the power value

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryPowerVal (ViSession handle, ViReal64 fVal[])*

Function Purpose Query the power value.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *fVal[]*—Power value.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.6 Universal Commands

3.6.1 [Store/Recall]

Screen capture

Function Prototype *ViStatus _VI_FUNC AV3680A_StoreScreen(ViSession handle, ViChar* pVal)*

Function Purpose Screen capture, and save it as a JPG file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete a single picture file

Function Prototype *ViStatus _VI_FUNC AV3680A_DelPictureFile(ViSession handle, ViChar* pVal)*

Function Purpose Delete a single picture file.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *pVal*—File name.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Delete all the picture files

Function Prototype *ViStatus _VI_FUNC AV3680A_DelAllPictureFile(ViSession handle)*

Function Purpose Delete all the picture files.

Input Parameter *handle*—Instrument communication handle.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the store location

Function Prototype *ViStatus _VI_FUNC AV3680A_SetFileLocation (ViSession handle, ViUInt32 nVal)*

Function Purpose Set the store location.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Stoere location, corresponding relation is:

0: U disc

1: SD card

2: Inside

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the store location

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryFileLocation (ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the store location.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Store location, corresponding relation is:

0: U disc

1: SD card

2: Inside

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

3.6.2 [System]

Set the measurement mode

Function Prototype *ViStatus _VI_FUNC AV3680A_SetInstSel (ViSession handle, ViUInt32 nVal)*

Function Purpose Set the measurement mode.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Measurement mode, corresponding relation is:

0: Cable & antenna measurement

1: Power measurement

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the measurement mode

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryInstSel(ViSession handle,*

ViUInt32 nVal[])

Function Purpose Query the measurement mode.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Measurement mode, corresponding relation is:

0: Cable & antenna measurement

1: Power measurement

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the available measurement mode

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryInstCatalog(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the available measurement mode.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Available measurement mode, corresponding relation is:

0: No measurement mode available

1: Only cable & antenna measurement mode available

2: Only power measurement mode available

3: Both cable & antenna measurement and power measurement mode available

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set GPS On or Off

Function Prototype *ViStatus _VI_FUNC AV3680A_SetGPSOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set GPS On or Off.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—GPS On or Off, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query GPS On or Off

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryGPSOn(ViSession handle, ViBoolean bVal[])*

Function Purpose Query GPS On or Off

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—GPS On or Off, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

GPS cold start

Function Prototype *ViStatus _VI_FUNC AV3680A_GPSReset(ViSession handle)*

Function Purpose GPS cold start, which may accelerate the startup by new GPS if no GPS data is available for a long period in places with very poor signal.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query GPS receiver state

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryGPSReceiveState(ViSession handle, ViBoolean bVal[])*

Function Purpose Query GPS receiver state

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—GPS receiver state, corresponding relation is:

0: the receiver has no data.

1: the receiver has data.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query GPS state

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryGPSState(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query GPS state.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—GPS receiver state, corresponding relation is:

0: no positioning

1: non differential positioning

2: differential positioning

3: ineffective PPS

4: estimate

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query GPS data

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryGPSData(ViSession handle, ViChar strVal[])*

Function Purpose Query GPS data.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *strVal []*—GPS data character string.

Return the current GPS data in the following format: < long.>,< lat. >,< alt.>,<time UTC>. For example: 38 28’11.22” N,122 42’13.23” W,152,06/28/2010 23:35:38. Return ”--,--,--,--\n” if no data available. .

The end of character string contains a line break (“\n”) and a stop character (“\0”).

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the sleep time

Function Prototype *ViStatus _VI_FUNC AV3680A_SetSleepTime(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the sleep time.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Value of sleep time, in min, the range of which is 1~600min.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the sleep time

Function Prototype *ViStatus _VI_FUNC AV3680A_QuerySleepTime(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the sleep time.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Value of sleep time, in min.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the shutdown time

Function Prototype *ViStatus _VI_FUNC AV3680A_SetShutdownTime(ViSession handle, ViUInt32 nVal)*

Function Purpose Set the shutdown time.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *nVal*—Value of shutdown time, in min, the range of which is15~2000min.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the shutdown time

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryShutdownTime(ViSession handle, ViUInt32 nVal[])*

Function Purpose Query the shutdown time.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]*—Value of shutdown time, in min.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On or Off of sleep

Function Prototype *ViStatus _VI_FUNC AV3680A_SetAutoSleepOn(ViSession handle, ViBoolean bVal)*

Function Purpose Set the On or Off of sleep.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—On or Off of sleep, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On or Off of sleep

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryAutoSleepOn(ViSession handle, ViBoolean bVal[])*

Function Purpose Query the On or Off of sleep.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—On or Off of sleep, corresponding relation is:

0: Off

1: 开

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Set the On or Off of automatic shutdown

Function Prototype *ViStatus _VI_FUNC AV3680A_SetAutoShutdownOn (ViSession handle, ViBoolean bVal)*

Function Purpose Set the On or Off of automatic shutdown.

Input Parameter *handle*—Instrument communication handle.

Input Parameter *bVal*—On or Off of automatic shutdown, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the On or Off of automatic shutdown

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryAutoShutdownOn (ViSession handle, ViBoolean bVal[])*

Function Purpose Query the On or Off of automatic shutdown.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *bVal[]*—On or Off of automatic shutdown, corresponding relation is:

0: Off

1: On

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.

Query the instrument internal temperature

Function Prototype *ViStatus _VI_FUNC AV3680A_QueryTemperature(ViSession handle,*

ViUInt32 nVal[])

Function Purpose Query the instrument internal temperature.

Input Parameter *handle*—Instrument communication handle.

Output Parameter *nVal[]—*Celsius temperature.

Return Value Command execution state, the normal value is VI_SUCCESS, otherwise other value is returned.