



# PicoVNA<sup>®</sup>

## Vector Network Analyzer

### Programmer's Guide

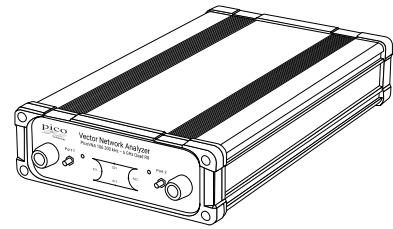
# Contents

1 Welcome .....	4
1 Software license conditions .....	4
2 Trademarks .....	5
3 System requirements .....	6
4 Contact information .....	6
2 Programming with PicoVNA .....	7
1 Installing the software .....	7
3 API functions .....	9
1 Command summary .....	9
2 Communications .....	11
1 FND() - discover instrument .....	11
2 FNDSN() - discover instrument by hardware serial number .....	12
3 CloseVNA() - close instrument .....	13
4 ResetErr() - Reset serial communications error flag .....	14
3 Calibration .....	15
1 SelectKit() - create kit [dialog] .....	15
2 SetKit() - create kit .....	16
3 SetFreqPlan() - set frequency plan .....	18
4 SetCWmode() - set CW sweep plan .....	19
5 SelectCal() - calibrate and set frequency plan [dialog] .....	20
6 MeasCal() - measure calibration standard .....	21
7 AppCal() - apply calibration .....	23
8 SelectP1dB() - calibration and measurement [dialog] .....	24
9 SelectAMPM() - AM-PM calibration and measurement [dialog] .....	25
10 DoP1dBCal() - P1dB Calibration .....	26
11 DoAMPMCal() - AM-PM Calibration .....	27
12 SelectMixerFrm() - select mixer [dialog] .....	28
13 SetRXQ() - set quiescent point of receiver .....	29
4 Measurements .....	30
1 SetTrig() - set sweep trigger mode .....	30
2 PulseTrigOut() - pulse the trigger output .....	31
3 Measure() - measure one sweep (S11, S21, S11+S21, or "All" using current calibration) .....	32
4 DoP1dBMeas() - measure P1dB .....	33
5 DoAMPMMeas() - measure AM-PM .....	34
5 Signal processing .....	35
1 SetEnhance() - set enhancement parameters .....	35
2 SetRef() - set reference plane .....	37
3 SaveToMem() - save measurement to memory .....	38
4 AppMemMath() - apply memory math .....	39

5 SetSysZo() - set system impedance .....	40
6 ZConversion() - impedance conversion utility .....	41
6 Get processed data .....	42
1 GetData() - get data .....	42
2 GetMem() - get memory .....	43
3 FndPt() - find data point .....	44
4 SetLimits() - set pass/fail limits .....	45
5 TestLimits() - pass/fail measurement .....	46
7 Get info .....	47
1 GetInfo() - get instrument/cal information .....	47
8 Data storage .....	49
1 SaveKit() - save cal kit .....	49
2 SelectSaveMeas() - save measurement .....	50
3 SaveCal() - save status and calibration .....	51
9 Data retrieval .....	52
1 LoadCal() - load status and calibration .....	52
10 Miscellaneous .....	53
1 InitVar() - initialize all variables .....	53
2 SelectSigGen() - set to signal generator mode [dialog] .....	54
3 SetSigGen() - set to signal generator mode .....	55
4 SelectPhaseMeter() - measure phase difference interactively .....	56
5 InitialisePhaseMeter() - set frequency and bandwidth .....	57
6 MeasurePhase() - run the phase meter .....	58
7 SelectSOT() - save on trigger [dialog] .....	59
8 SOTMeasure() - do save on trigger measurement .....	60
9 SOT_CopySweep() - copy to trace .....	61
10 SOT_SaveData() - save sweep to data array .....	62
11 DLLVer() - get DLL program version .....	63
11 Diagnostics .....	64
1 InsDiag() - run diagnostics tests [dialog] .....	64
4 Glossary .....	65
Index .....	66

# 1 Welcome

The **PicoVNA Vector Network Analyzers** from Pico Technology are compact, high-performance measuring instruments. A VNA works by driving a swept sine wave test signal into one port of a [DUT](#) and measuring the reflected (and optionally the transmitted) signals. The signal may then be injected into the other port of the DUT for a further set of measurements. The VNA then uses these measurements to calculate the DUT's S-parameters.



This manual describes the operation of the **PicoVNA 2** and **PicoVNA 3** libraries available to support the PicoVNA 106 and PicoVNA 108 Vector Network Analyzers respectively. The key intention of the libraries is to provide remote automation under standard programming environments such as C, C++, C# or specialist test and measurement, scientific and math environments such as National Instruments LabVIEW and MathWorks MATLAB.

[PicoVNA product web page >>](#)

## 1.1 Software license conditions

The material contained in this release is licensed, not sold. Pico Technology Limited grants a license to the person who installs this software, subject to the conditions listed below.

**Access.** The licensee agrees to allow access to this software only to persons who have been informed of these conditions and agree to abide by them.

**Copyright.** Pico Technology Ltd. claims the copyright of, and retains the rights to, all SDK materials (software, documents, etc.) except the example programs. You may copy and distribute SDK files without restriction, as long as you do not remove any Pico Technology copyright statements. The example programs may be modified, copied and distributed for the purpose of developing programs to collect data using Pico products.

**Liability.** Pico Technology and its agents shall not be liable for any loss, damage or injury, howsoever caused, related to the use of Pico Technology equipment or software, unless excluded by statute.

**Fitness for purpose.** As no two applications are the same, Pico Technology cannot guarantee that its equipment or software is suitable for a given application. It is your responsibility, therefore, to ensure that the product is suitable for your application.

**Mission-critical applications.** This software is intended for use on a computer that may be running other software products. For this reason, one of the conditions of the license is that it excludes use in mission-critical applications, for example life support systems.

**Viruses.** This software was continuously monitored for viruses during production, but you are responsible for virus-checking the software once it is installed.

**Support.** If you are dissatisfied with the performance of this software, please contact our technical support staff, who will try to fix the problem within a reasonable time. If you are still dissatisfied, please return the product and software to your supplier within 14 days of purchase for a full refund.

**Upgrades.** We provide upgrades, free of charge, from our website at [www.picotech.com](http://www.picotech.com). We reserve the right to charge for updates or replacements sent out on physical media.

## 1.2 Trademarks

*Pico Technology* is an internationally registered trademark of Pico Technology Ltd. *PicoVNA* and *PicoSDK* are registered trademarks of Pico Technology Ltd.

*Windows*, *Excel* and *Visual Basic for Applications* are registered trademarks or trademarks of Microsoft Corporation in the USA and other countries. *LabVIEW* is a registered trademark of National Instruments Corporation. *MATLAB* is a registered trademark of The MathWorks, Inc.

## 1.3 System requirements

To ensure that your PicoVNA Vector Network Analyzer operates correctly, you must have a computer with at least the minimum system requirements to run one of the supported operating systems, as shown in the following table. The performance of the analyzer will be better with a more powerful PC, and will benefit from a multicore processor.

Item	Specification
<b>Operating system</b>	Windows 7, 8 or 10 (32-bit and 64-bit versions)
<b>Processor, memory, free disk space</b>	As required by the operating system
<b>Ports</b>	<a href="#">USB 2.0</a> port

## 1.4 Contact information

Pico provides tailored technical and sales support to all customers. Contact us via our website, [www.picotech.com](http://www.picotech.com), or using the details below.

### United Kingdom global headquarters

Pico Technology  
James House  
Colmworth Business Park  
St. Neots  
Cambridgeshire  
PE19 8YP  
United Kingdom

☎: +44 (0) 1480 396 395  
✉: [support@picotech.com](mailto:support@picotech.com)  
✉: [sales@picotech.com](mailto:sales@picotech.com)

**UK office hours:**  
09:00 to 17:00 (UK time)  
Monday to Friday

### North America regional office

Pico Technology  
320 N Glenwood Blvd  
Tyler  
TX 75702  
United States

☎: +1 800 591 2796  
✉: [support@picotech.com](mailto:support@picotech.com)  
✉: [sales@picotech.com](mailto:sales@picotech.com)

**North America office hours:**  
09:00 to 17:00 (Central time)  
Monday to Friday

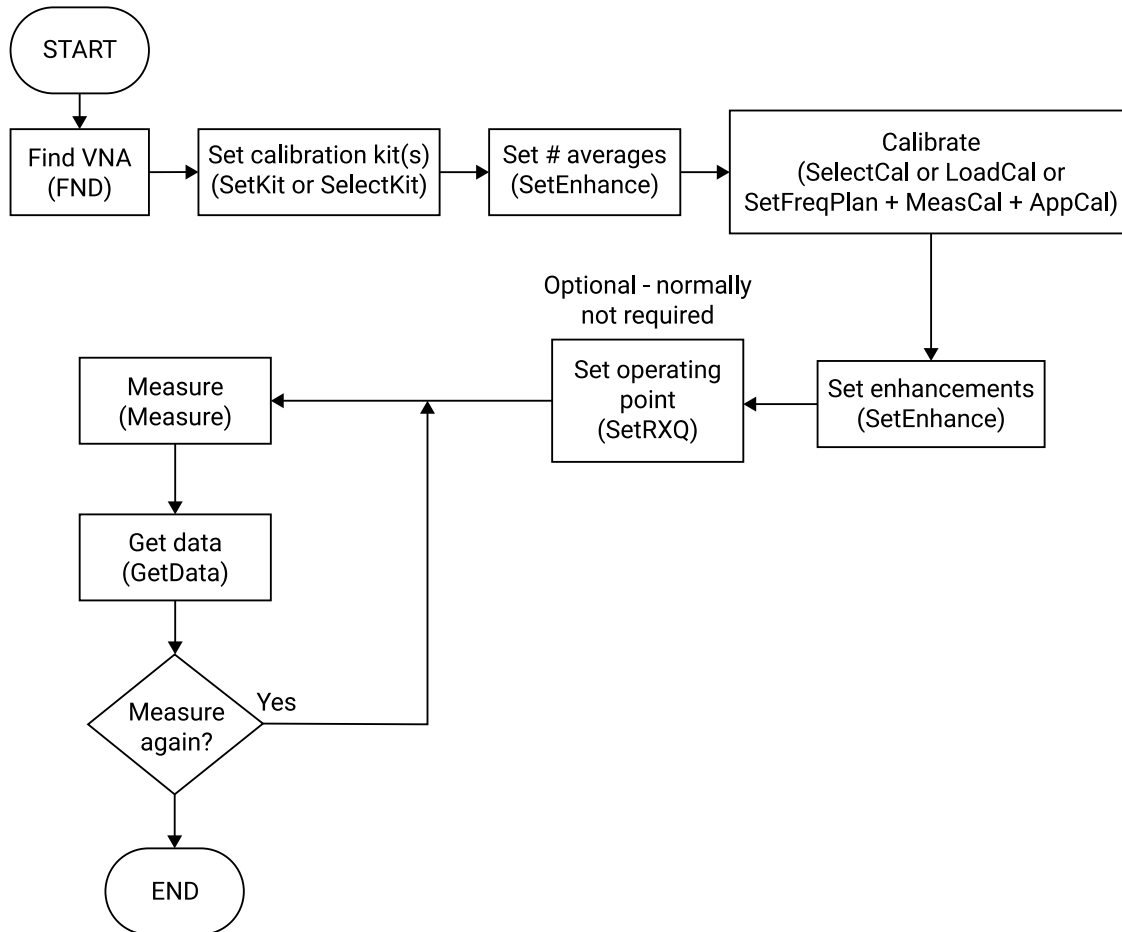
### Asia-Pacific regional office

Pico Technology  
Room 2252, 22/F, Centro  
568 Hengfeng Road  
Zhabei District  
Shanghai 200070  
PR China

☎: +86 21 2226-5152  
✉: [pico.asia-pacific@picotech.com](mailto:pico.asia-pacific@picotech.com)

## 2 Programming with PicoVNA

The diagram below shows the steps to carry out a measurement. It is simplified and incorporates the minimum number of steps to perform a measurement.



### 2.1 Installing the software

1. Download the latest PicoVNA software installer from [www.picotech.com/downloads](http://www.picotech.com/downloads):  
Use the PicoVNA 2 software for the PicoVNA 106.6 GHz VNA  
Use the PicoVNA 3 software for the PicoVNA 108.8.5 GHz VNA
2. Run the installer.
3. In case of difficulty, please consult the [PicoVNA Quick Start Guide](#) for further help.

The control library is a 32-bit Windows COM [DLL](#):

PicoControl2.dll is supplied with the PicoVNA 2 software  
PicoControl3.dll is supplied with the PicoVNA 3 software

Installation of the PicoVNA 2 or 3 software automatically installs and registers the relevant COM class on the PC:

PicoControl2.PicoVNA\_2 for the PicoVNA 2 software  
PicoControl3.PicoVNA\_3 for the PicoVNA 3 software

**DLL location**

The PicoVNA software installer places the location of the DLL on the Windows path, which allows most development tools to find it. If your development tool needs to know the location of this DLL, you may need to modify the tool's search path to point to it.

**Using multiple devices**

The PicoVNA SDKs support the use of up to three devices at a time. To enable this, two additional DLLs are available (`picovna_1.dll` and `picovna_2.dll`). They contain similar functions to those in `PicoControl2.dll` and `PicoControl3.dll`, but with a suffix of `_1` or `_2` respectively appended to the function names. Contact Pico Technical Support to obtain the extra DLLs.

Example code for operating three devices is available on [Pico's GitHub organization page](#), in the [picosdk-matlab-picovna-vector-network-analyzer-toolbox](#) repository.

**Mixer measurements (PicoVNA 108 only)**

If you wish to carry out mixer measurements with a GPIB-connected external instrument it is very likely you will need to install the Keysight® IO Libraries Suite. Please see [www.keysight.com](http://www.keysight.com) for details. No additional software is required for communicating with USB-connected instruments.



## 3 API functions

### 3.1 Command summary

Command	Description	Input	Output
<a href="#">AppCal()</a>	Apply the calibration	CalType	String: "OK" or "Error"
<a href="#">AppMemMath()</a>	Apply vector math to the current measurement	Para, Func	String: "OK" or "Error"
<a href="#">CloseVNA()</a>	Close the USB connection	None	String: "OK" or "Error"
<a href="#">DLLVer()</a>	Query the current <a href="#">DLL</a> version	None	String: <dll version>
<a href="#">DoP1dBCal()</a>	Perform a <a href="#">P1dB</a> calibration	Loss1, Loss2, Ftest	String: "OK" or "Error"
<a href="#">DoAMPMCal()</a>	Perform an <a href="#">AM to PM</a> calibration	Loss1, Loss2, Ftest	String: "OK" or "Error"
<a href="#">DoP1dBMeas()</a>	Measure P1dB	None	String: <P1dB value> or <error message>
<a href="#">DoAMPMMeas()</a>	Measure AM to PM	Ptest	String: <AM PM value> or <error message>
<a href="#">FND()</a>	Find instrument	None	Integer, serial port
<a href="#">FndPt()</a>	Read data (marker function)	Freq, Para, MeasType, Func	String: <freq, pk or min value, bandwidth, freq index> or "Error"
<a href="#">GetData()</a>	Read measured data	Para, MeasType, Pnt	String: <freq, value> or "Error"
<a href="#">GetInfo()</a>	Read instrument status	Para	String: <value>, "Error"
<a href="#">GetMem()</a>	Read memory data	Para, MeasType, Pnt	String: <freq, value> or "Error"
<a href="#">InitialisePhaseMeter()</a>	Set phase meter parameters	Fq, B	String: "OK" or error message
<a href="#">InitVar()</a>	Initialize all variables	None	String: "OK"
<a href="#">InsDiag()</a>	Perform diagnostic tests	None	String: "Tests completed" or "Tests not performed"
<a href="#">LoadCal()</a>	Load a calibration and status from disk	FileName or "?"	String: "OK" or "Error"
<a href="#">MeasCal()</a>	Measure a calibration standard	CalType, Standard	String: "OK" or "Error"
<a href="#">Measure()</a>	Perform a single sweep measurement	Para	String: "OK" or "Error"
<a href="#">MeasurePhase()</a>	Perform a phase measurement	Mode, PhRef	Comma-separated string
<a href="#">PulseTrigOut()</a>	Pulse the trigger output	None	String: "OK" or "Error"
<a href="#">ResetErr()</a>	Reset communication error flag	None	None
<a href="#">SaveCal()</a>	Save the current calibration and status to disk	FileName or "?"	String: "OK" or "Error"
<a href="#">SaveKit()</a>	Calibration kit [Note: Thru data only saved if kit is on port 1]	FileName, Port	String: "OK" or "Error"
<a href="#">SaveToMem()</a>	Save measured data to memory	Para	String: "OK" or "Error"
<a href="#">SelectAMPM()</a>	Measure AM to PM	None	None (displays window)

<a href="#">SelectCal()</a>	Calibrate the instrument	"?" or FileName	String: "OK" or "Error" (displays window)
<a href="#">SelectKit()</a>	Create, load or save a calibration kit	"?" or FileName, Port	String: "OK" or "Error" (displays window)
<a href="#">SelectMixerFrm()</a>	Mixer measurement	None	None (displays window)
<a href="#">SelectP1dB()</a>	Measure P1dB	None	None (displays window)
<a href="#">SelectPhaseMeter()</a>	Measure phase difference interactively	None	String: "OK" (displays window)
<a href="#">SelectSaveMeas()</a>	Save measured data to disk	None	None (displays window)
<a href="#">SelectSigGen()</a>	Set up the instrument as a synthesized signal generator	None	None (displays window)
<a href="#">SelectSOT()</a>	Display Save on Trigger dialog form to examine and/or save captured sweep data	None	String: "OK" or "Error" (displays window)
<a href="#">SOTMeasure()</a>	Carry out a preset number of sweep measurements and store all sweep measurements	Number of sweeps to capture, trigger mode (rise or falling edge only), measurement parameter, time out in milliseconds	String: "OK" or "Error"
<a href="#">SOT_CopySweep()</a>	Copy captured sweep to live or memory trace array so that it can be downloaded	Sweep, array	String: "OK" or "Error"
<a href="#">SOT_SaveData()</a>	Copy Save on Trig measurements to disk file(s)	File name, Para, Format, Mode, Frq	String: "OK" or "Error"
<a href="#">SetEnhance()</a>	Set measurement enhancement	Para, k	String: "OK" or "Error"
<a href="#">SetFreqPlan()</a>	Set the <a href="#">frequency plan</a>	Fstart, Fstep, I, P, BW	String: "OK" or "Error"
<a href="#">SetKit()</a>	Create a calibration kit	Name, Ksex, Koff, Koff2, CF0, CF1, CF3, Sind, OffLoss1, OffLoss2, Port, MData, FileName1, Tdata, FileName2	String: "OK" or "Error"
<a href="#">SetLimits()</a>	Set pass / fail limits	Segment, Para, FreqLow, FreqHigh, ValMin, ValMax	String: "Error", "OK" or data string
<a href="#">SetSigGen()</a>	Set up the instrument as a synthesized signal generator	Freq, P	String: "OK" or "Error"
<a href="#">SetRef()</a>	Set measurement <a href="#">reference plane</a>	Para, k	String: "OK", "Error" or "No cal"
<a href="#">SetRXQ()</a>	Set quiescent point of the receiver	None	String: "OK" or "Error"
<a href="#">SetTrig()</a>	Set the trigger mode	Mode	String: "OK" or "Error"
<a href="#">TestLimits()</a>	Do a pass/fail test on the current measured data against set pass/fail limits	Para, FreqLow, FreqHigh, MeasType	String: "Error", "Pass", or string data with failed data
<a href="#">ZConversion()</a>	Convert a set of s-parameters from one base <a href="#">impedance</a> to another	ZoOld, ZoNew, S11r, S11i, S21r, S21i, S12r, S12i, S22r, S22i	String: data string or "Error"
<a href="#">SetSysZo()</a>	Set the system impedance to a value other than 50 $\Omega$	SysZo, ExMatch	String: "OK" or "Error"

## 3.2 Communications

### 3.2.1 FND() - discover instrument

Public Function FND() As Integer

#### Typical usage

This function is used to find the instrument connected to the PC.

```
variable = FND
```

#### Action

Searches all available serial ports to find the instrument. If the instrument is found, the function reads factory data from the instrument's [EEPROM](#). This takes about three seconds to complete.

Parameter passed	Description	Value
None		

#### Returns

Serial number of the instrument detected or 0 if no instrument is found.

Return values	Type	Executed correctly	Error
<0 to 9999>	Integer	<1 to 9999>	<0>

Alternatively, you can find a specific instrument by its hardware serial number using [FNSN\(\)](#).

### 3.2.2 FNDSN() - discover instrument by hardware serial number

```
Public Function FNDSN(
    ByVal SN as string
) As Integer
```

#### Typical usage

This function is used to find a specific instrument connected to the PC:

```
variable = FNDSN("A3VIAUAH")
```

#### Action

Searches all available USB ports to find the instrument with a matching hardware serial number. If the instrument is found, the function reads factory data from the instrument's [EEPROM](#). This takes about 3 seconds to complete. No parameter is passed in the call.

You can find out the hardware serial number of a device by requesting the "HWSN" parameter from [GetInfo\(\)](#). Note that this has a different value from the serial number returned by [FND\(\)](#) and [FNDSN\(\)](#).

Parameter passed	Description	Value
SN	Hardware serial number	A string such as "A3VIAUAH".

#### Returns

Serial number of the instrument detected or 0 if no instrument is found.

Return values	Type	Executed correctly	Error
<0 to 9999>	Integer	<1 to 9999>	<0>

### 3.2.3 CloseVNA() - close instrument

Public Function CloseVNA() As String

#### Typical usage

This function is used to close the instrument connected to the PC.

```
variable = CloseVNA
```

#### Action

Closes the USB port currently assigned to the instrument.

Parameter passed	Description	Value
None		

#### Returns

"OK" if currently open USB port is successfully closed. If no USB port currently open the value returned is "Error".

Return values	Type	Executed correctly	Error
"OK", "Error"	String	"OK"	"Error"

### 3.2.4 ResetErr() - Reset serial communications error flag

```
Public Sub ResetErr()
```

**Typical usage**

This routine is used to reset the error flag.

**Action**

Resets the serial communications error flag. No parameters are passed in the call.

Parameter passed	Description	Value
None		

**Returns**

Nothing

### 3.3 Calibration

#### 3.3.1 SelectKit() - create kit [dialog]

```
Public Function SelectKit(
    ByVal FileName As String,
    ByVal port As Integer
) as String
```

**Typical usage**

This routine is used to create or load a [calibration kit](#).

```
<variable> = SelectKit(<FileName>, <port>)
```

**Action**

If you pass a “?” instead of a file name, this function displays a dialog that allows the user to select (load from disk) or create and save a calibration kit. On the other hand, if a valid full path file name is passed (of the calibration kit to be loaded) together with either “1” for port 1, or “2” for port 2, the kit is loaded and applied without a dialog.

Parameter passed	Description	Comments
FileName, Port or “?”	Full path file name (of calibration kit), test port (“1” or “2”) or “?”	If a “?” is passed, a dialog is displayed

**Returns**

Return values	Type	Executed correctly	Error
“OK” or “Error”	String	“OK”	“Error”

Example calibration kit file for a female kit with load and [through](#) standard data:

```

"MyKit_female" ← Kit name
#FALSE# ← Flag: True for male kit
.005,1.66782043871207E-11,.0,.0,2.5E-15,2.9E-25,-1E-35,2.4E-46 } Short offset(m), Short offset time (s),
0 } Open offset (m), offset time (s), and open
0,0 ← Short standard inductance standard capacitance coeffs
{
# TRUE # ← Short and Open offset loss
.3,-.0028035,1.7535E-04
40.3,-.0021602,4.3693E-04
80.3,-.0015778,7.6176E-04
.
.
.
7960.3,-.002981,-.0067849
8000.3,-.0031369,-.0064241
}
#TRUE#
.3,0,0,.9999904,-1.005996E-04,.9999904,-1.005996E-04,0,0
40.3,0,0,.9997978,-.0135121,.9997978,-.0135121,0,0
80.3,0,0,.999481,-2.691995E-02,.999481,-2.691995E-02,0,0
.
.
.
7960.3,0,0,-.8891715,-.4541592,-.8891715,-.4541592,0,0
8000.3,0,0,-.8951797,-.4421901,-.8951797,-.4421901,0,0
EOF

```

In the above, the Short and Open offset refer to the Open and Short [reference plane](#) distances. The offset times are the equivalent transit time in free space for this plane distance.

### 3.3.2 SetKit() - create kit

```
Public Function SetKit(  
    ByVal name      As String,  
    ByVal Ksex      As String,  
    ByVal Koff      As Variant,  
    ByVal Koff2     As Variant,  
    ByVal CF0       As Variant,  
    ByVal CF1       As Variant,  
    ByVal CF2       As Variant,  
    ByVal CF3       As Variant,  
    ByVal SIND      As Variant,  
    ByVal OffLoss1  As Variant,  
    ByVal OffLoss2  As Variant,  
    ByVal port      As Integer,  
    ByVal MData     As String,  
    ByVal FileName1 As String,  
    ByVal Tdata     As String,  
    ByVal FileName2 As String  
) As String
```

#### Typical usage

```
variable =  
    SetKit(name, Ksex, Koff, Koff2, CF0, CF1, CF2, CF3, SIND, OffLoss1, OffLoss2, port, MData,  
    ta, FileName1, Tdata, FileName2)
```



**Action**

Creates a [calibration kit](#) using the parameters passed. See table below for the type and format of data required. Note that Thru data is only loaded if the kit is to be associated with Port 2 of the instrument.

Parameter passed	Description	Value	Unit
Name	Cal kit name		
Ksex	Cal kit connector sex	"m" or "f"	
Koff	Cal kit's Open standard offset		m
Koff2	Cal kit's Short standard offset		m
CF0	Open standard capacitance coefficient		F
CF1	Open standard capacitance coefficient		F
CF2	Open standard capacitance coefficient		F
CF3	Open standard capacitance coefficient		F
SIND	Short standard inductance		H
OffLoss1	Short offset loss	Typically around 2	GΩ/s
OffLoss2	Open offset loss	Typically around 2	GΩ/s
Port	Port (1 or 2) to which kit applies	1 or 2	
MData	Flag indicates if load data is available	"True" or "False"	
FileName1	Path and file name of Load data file		
TData	Flag indicates if Thru data is available	"True" or "False"	
FileName2	Path and file name of Thru data file		

**Returns**

Return values	Type	Executed correctly	Error
"OK" or "Error"	String	"OK"	"Error"

**Important:** [Through](#) adaptor data is only loaded and saved when the kit is associated with **Port 1** of the instrument. A kit with Thru data may be loaded to Port 2 but the Thru data will be ignored, therefore, if the kit is subsequently resaved, the Thru data will be lost.

### 3.3.3 SetFreqPlan() - set frequency plan

```
Public Function SetFreqPlan(
    ByVal Fstart As Variant,
    ByVal Fstep As Variant,
    ByVal I As Variant,
    ByVal P As Variant,
    ByVal B As Variant
) As String
```

#### Typical usage

This routine is used to set the [frequency plan](#) of the instrument. Typically, the format used is as follows:

```
variable = SetFreqPlan(start_frequency, frequency_step, no_of_points,
    power_level, bandwidth)
```

#### Action

Sets the frequency sweep.

Parameter passed	Units	Min Value	Max Value	Step size
start_frequency	MHz	0.3	6080 or 8500	
frequency_step	MHz	0.00001 (10 Hz)	-	
no_of_points (I)		51,101, 201, 401, 801, 1001, 1024, 2001, 2048, .... , 10001		
power_level	dBm	-20	+6 (PicoVNA 106) +10 (PicoVNA 108)	0.1
bandwidth	Hz	10	140 000	

The sweep is set by passing the start frequency and step size. The step size is calculated by subtracting the start frequency from the stop frequency and dividing by the (number of points – 1).

$$F_{stop} = F_{start} + F_{step} \times (N_{points} - 1) \leq 6080 \text{ or } 8500 \text{ MHz}$$

#### Returns

Return values	Type	Executed correctly	Error
"OK" or "Error"	String	"OK"	"Error"

**Note:** If a valid [calibration](#) exists, setting a new frequency plan (different from that used during calibration) will cause a new set of (interpolated) error terms to be generated so that a new calibration need not be carried out. Note that this process will delete any measurements saved to memory. Note further that in order to obtain the best instrument capability a new calibration should be performed.

### 3.3.4 SetCWmode() - set CW sweep plan

```
Public Function SetCWmode(
    ByVal F As Variant,
    ByVal pts As Variant,
    ByVal Po As Variant,
    ByVal Td As Variant
) As String
```

#### Typical usage

This routine is used to set the [CW](#) sweep plan (time sweep) of the instrument. Typically, the format used is as follows:

```
variable = SetCWmode(frequency, no_of_points, power_level,
    time_between_points)
```

#### Action

Sets the CW mode sweep.

Parameter passed	Units	Min Value	Max Value	Resolution
frequency	MHz	0.3	6080 (PicoVNA 106) 8500 (PicoVNA 108)	*
no_of_points	-	2	10001	1
power_level	dBm	-20	0 (PicoVNA 106) 10 (PicoVNA 108)	0.1
time_between_points	ms	0.5	65	0.001

\*Frequency resolution:

```
0.3 MHz to 62.5 MHz:    0.1 Hz
62.5 MHz to 3 GHz:     10 Hz or better (PicoVNA 106)
                       5 Hz or better (PicoVNA 108)
3 GHz to 6 GHz:       20 Hz (PicoVNA 106)
3 GHz to 8.5 GHz:    10 Hz (PicoVNA 108)
```

#### Returns

Return values	Type	Executed correctly	Error
"OK" or "Error"	String	"OK"	"Error"

### 3.3.5 SelectCal() - calibrate and set frequency plan [dialog]

```
Public Function SelectCal(
    ByVal FileName As string
) as String
```

#### Typical usage

This routine is used to set the [frequency plan](#) and [calibrate](#) the instrument.

#### Action

If a "?" string is passed then this function displays a dialog that allows the user to set the frequency plan (start, stop and number of frequency points) and complete calibration of the instrument. If a valid full path file name holding a calibration is passed, the calibration will be loaded without displaying a dialog.

Parameter passed	Description	Value
FileName or "?"	Full path file name (of calibration file) or "?"	If a "?" is passed a dialog is displayed

#### Returns

Return values	Type	Executed correctly	Error
"OK" or "Error"	String	"OK"	"Error"

**Note:** If a valid calibration exists, setting a new frequency plan (different from that used during calibration) only will cause a new set of (interpolated) error terms to be generated so that a new calibration need not be carried out. Note that this process will delete any measurements saved to memory. Note further that in order to obtain the best instrument capability a new calibration should be performed whenever the sweep parameters change.

### 3.3.6 MeasCal() - measure calibration standard

```
Public Function MeasCal(
    ByVal CalType As Variant,
    ByVal Standard As Variant
) As String
```

#### Typical usage

```
variable = MeasCal(CalType, Standard)
```

#### Action

Measures a calibration standard (used during the [calibration](#) procedure). The parameters passed are described below.

Parameter passed	Description	Value
CalType	Calibration type	"S11", "S21", "S11+S21", "All", "Alln", "Allut"
Standard	Calibration standard	"Open", "Short", "Load", "Thru", "Isolation", "OpenP1", "OpenP2", "ShortP1", "ShortP2", "ShortP1OpenP2", "OpenP1ShortP2", "LoadP1", "LoadP2", "LoadP1ShortP2" or "LoadP2ShortP1"

**Note!** Calibration types are as follows:

"S11"	1 port correction
"S21"	frequency response (with optional isolation) correction
"S11+S21"	1 port correction, with frequency response and isolation correction and source match correction
"All"	Full 12-term correction for insertable device (2 cal kits needed)
"Alln"	Full 12-term correction for non-insertable device (only 1 cal kit)
"Allut"	Unknown <a href="#">through</a> calibration (8-term correction) for non-insertable device (only 1 cal kit)

S11: Calibration standards for one-port measurement are as follows:

"Load"	loads on port 1
"Open"	opens on port 1
"Short"	shorts on port 1

S21: Calibration standards for through measurement and normalization measurement are as follows:

"Thru"	connect ports 1 and 2
"Isolation"	terminate port 1

S11+S21: Calibration standards for 2-port device measurement are as follows:

"Load"	loads on port 1 (also measures isolation term)
"Open"	opens on port 1 (also measures isolation term)
"Short"	shorts on port 1 (also measures isolation term)
"Thru"	connect ports 1 and 2

This calibration method uses the various isolation measurements in order to attempt to model the termination-dependent isolation term of the instrument. This approach is similar to that described in the *User's Guide* as the "enhanced isolation calibration".

All: Calibration standards for the *insertable* device (2 cal kits needed) are as follows:

"Load"	loads on port 1 and 2
"Open"	opens on port 1 and 2
"Short"	shorts on port 1 and 2
"Thru"	connect ports 1 and 2
"Isolation0"	terminate ports 1 and 2 [conventional isolation calibration]
"Isolation1"	short on port 1 and open on port 2 [enhanced isolation calibration]

Alln: Calibration standards for the *non-insertable* device (only 1 cal kit) are as follows:

"LoadP1ShortP2"	load on port 1, short on port 2 (also measures isolation term)
"LoadP2ShortP1"	load on port 2, short on port 1 (also measures isolation term)
"Thru"	connect ports 1 and 2 using calibrated through standard

Reflection standards are:

"ShortP1OpenP2"	short circuit on port 1, open circuit on port 2 (also measures isolation term)
"OpenP1ShortP2"	open circuit on port 1, short circuit on port 2 (also measures isolation term)

This calibration method uses the various isolation measurements in order to attempt to model the termination-dependent isolation term of the instrument. This approach is described in the *User's Guide* as the "enhanced isolation calibration".

Allut: Calibration standards for the *non-insertable* device using the unknown through calibration (only 1 cal kit) are as follows:

"LoadP1ShortP2"	load on port 1, short on port 2 (also measures isolation term)
"LoadP2ShortP1"	load on port 2, short on port 1 (also measures isolation term)
"Thru"	Connect ports 1 and 2 using reciprocal through adaptor

Reflection standards are:

"ShortP1OpenP2"	short circuit on port 1 and open circuit on port 2 (also measures isolation term)
"OpenP1ShortP2"	open circuit on port 1 and short circuit on port 2 (also measures isolation term)

This calibration method uses the various isolation measurements in order to attempt to model the termination-dependent isolation term of the instrument. This approach is described in the *User's Guide* as the "enhanced isolation calibration".

### Returns

Return values	Type	Executed correctly	Not executed
"OK" or "Error"	String	"OK"	"Error"

### 3.3.7 AppCal() - apply calibration

```
Public Function AppCal(
    ByVal CalType As Variant
) As String
```

#### Typical usage

```
variable = AppCal(CalType)
```

#### Action

Applies the [calibration](#) (used after all calibration standards, i.e. short, open and load, have been measured). The parameter passed is described in the table below.

Parameter passed	Description	Value
CalType	Calibration type	"S11", "S21", "S11+S21", "All", "Alln"

#### Returns

Return values	Type	Executed correctly	Not executed
"OK" or "Error"	String	"OK"	"Error"

**Note!** Only use with the [MeasCal\(\)](#) commands. Not after using [SelectCal\(\)](#) or [LoadCal\(\)](#).

### 3.3.8 SelectP1dB() - calibration and measurement [dialog]

```
Public Sub SelectP1dB()
```

**Typical usage**

This function is used to perform [P1dB](#) calibration and measurements. Typically implemented as follows:

```
Call SelectP1dB
```

or:

```
SelectP1dB
```

**Action**

A dialog is displayed that allows the user to carry out P1dB measurements interactively. No parameters are passed.

Parameter passed	Description	Value
None		

**Returns**

Nothing



### 3.3.9 SelectAMPM() - AM-PM calibration and measurement [dialog]

Public Sub SelectAMPM()

#### Typical usage

Used to perform an [AM to PM](#) calibration. Typically implemented as follows:

```
Call SelectAMPM
```

or:

```
SelectAMPM
```

#### Action

A dialog is displayed that allows the user to carry out AM to PM measurements interactively. No parameters are passed.

Parameter passed	Description	Value
None		

#### Returns

Nothing

### 3.3.10 DoP1dBCal() - P1dB Calibration

```
Public Function DoP1dBCal(
    ByVal Loss1 As Variant,
    ByVal Loss2 As Variant,
    ByVal Ftest As Variant
) As String
```

#### Typical usage

Used to perform a [P<sub>1dB</sub>](#) calibration. Typically implemented as follows:

```
variable = DoP1dBCal(Loss1, Loss2, Ftest)
```

#### Action

Performs a P1dB calibration. This is used prior to carrying out a P1dB measurement. No dialog form is displayed. The parameters passed are described in the table below.

**Note!** P<sub>1dB</sub> Calibration should be carried out without the input or output attenuators in place.

Parameter passed	Description	Value	Unit
Loss1	Input attenuation / loss		dB
Loss2	Output attenuation / loss		dB
Ftest*	Frequency at which the Cal is done	0.3 to 6080 (PicoVNA 106) 0.3 to 8500 (PicoVNA 108)	MHz

\* setting Ftest to 0 will cause calibration using sweep plan stored

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

### 3.3.11 DoAMPMSCal() - AM-PM Calibration

```
Public Function DoAMPMSCal(
    ByVal Loss1 As Variant,
    ByVal Loss2 As Variant,
    ByVal Ftest As Variant
) As String
```

#### Typical usage

Used to perform an [AM to PM](#) calibration. Typically implemented as follows:

```
variable = DoAMPMSCal(Loss1, Loss2, Ftest)
```

#### Action

Performs an AM to PM calibration at the specified frequency. The parameters passed are described in the table below.

**Note!** AM-PM Calibration should be carried out without the input or output attenuators in place.

Parameter passed	Description	Value	Unit
Loss1	Input attenuation / loss		dB
Loss2	Output attenuation / loss		dB
Ftest	Frequency at which the Cal is done	0.3 to 6080 (PicoVNA 106) 0.3 to 8500 (PicoVNA 108)	MHz

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

### 3.3.12 SelectMixerFrm() - select mixer [dialog]

```
Public Sub SelectMixerFrm()
```

**Typical usage**

**PicoVNA 108 only.** This function is used to perform a mixer measurement calibration. Typically implemented as follows:

```
Call SelectMixerFrm
```

or:

```
SelectMixerFrm
```

**Action**

A dialog is displayed that allows the user to carry out mixer measurements interactively. No parameters are passed.

Parameter passed	Description	Value
None		

**Returns**

Nothing

### 3.3.13 SetRXQ() - set quiescent point of receiver

Public Function SetRXQ() As String

#### Typical usage

```
variable = SetRXQ
```

#### Action

This function optimizes the quiescent point of the instrument's receiver. This function is not normally necessary unless the ultimate capability of the instrument is required. If used, it must be called immediately before performing a measurement sweep. No parameters are passed.

Parameter passed	Description	Value
None		

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

## 3.4 Measurements

### 3.4.1 SetTrig() - set sweep trigger mode

```
Public Function SetTrig(
    ByVal Para As String,
    ByVal T As variant '[PicoVNA 108 only]
) As String
```

#### Typical usage

**PicoVNA 108 only.** Sets the trigger mode. It is typically called as follows:

```
StringVariable = SetTrig(para) '[PicoVNA 106 version]
StringVariable = SetTrig(para, T) '[PicoVNA 108 version]
```

#### Action

Sets the [sweep trigger](#) mode to either internal (free running) or externally triggered. The parameter passed is described in the table below.

Parameter passed	Description	Value
para	Mode of trigger	"Free", "Positive", "Negative"
T	Timeout in milliseconds PicoVNA 108 version only	1 to 1 200 000 (1 ms to 20 mins)

"Free" Normal operation, free running sweep

"Positive" Sweep is synchronized to the rising edge of the external trigger signal

"Negative" Sweep is synchronized to the falling edge of the external trigger signal

External trigger signal specification

Parameter	Units	Min	Max
Low level	V	-0.5	0.8
High level	V	1.5	5.0
Input resistance	kΩ	10	

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

## 3.4.2 PulseTrigOut() - pulse the trigger output

Public Function PulseTrigOut() As String

### Typical usage

**PicoVNA 108 only.** Pulses the trigger output for 1  $\mu$ s. It is typically called as follows:

```
StringVariable = PulseTrigOut()
```

### Action

Pulses the trigger output high for a period of 1  $\mu$ s. If the trigger output is high before the command is called it is first set low for 1  $\mu$ s before pulsing high. No parameters are passed with the call.

### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

### 3.4.3 Measure() - measure one sweep (S11, S21, S11+S21, or "All" using current calibration)

```
Public Function Measure(  
    ByVal Para As Variant  
) As String
```

#### Typical usage

Executes a measurement sweep. It is typically called as follows:

```
variable = Measure(Para)
```

#### Action

Performs a single measurement sweep using the current [calibration](#). The parameter passed is described in the table below.

Parameter passed	Description	Value
Para	Measurements to be made	"S11", "S21", "S11+S21", "All", "Raw"*

**Note!** If the current calibration is not for the parameter passed, then an error will be returned. For example, if "S21" is passed, then the current calibration must be either an S21, S11+S21 or "All" calibration. Similarly, if "S12" is passed, then the current calibration must be an "All" or "Alln" calibration.

\* "Raw" value available for PicoVNA 108 only. If "Raw" is passed, the instrument will perform a sweep and collect only the raw receiver data.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"



### 3.4.4 DoP1dBMeas() - measure P1dB

Public Function DoP1dBMeas() As String

#### Typical usage

Used to perform a [P1dB](#) measurement.

```
variable = DoP1dBMeas
```

#### Action

Performs a  $P_{1dB}$  measurement. Before this function is used, a  $P_{1dB}$  calibration must be carried out at which time the losses at the input and output of the device under test can be set. No parameters are passed.

Parameter passed	Description	Value
None		

#### Returns

Return values	Type	Executed correctly	Not executed correctly
$P_{1dB}$ value, "No cal" or "Error"	String*	< $P_{1dB}$ Value>	"Error" or "No cal"

**\*Note:** If calibration was done with `Ftest` parameter set to 0, the frequency sweep plan will be used. In this case, the string returned will have the following format:

```
<1st frequency pt in MHz>, <P1dB in dBm>, ....., <last frequency pt in MHz>,
<P1dB in dBm>
```

### 3.4.5 DoAMPMeas() - measure AM-PM

```
Public Function DoAMPMeas(
    ByVal Ptest As Variant
) As String
```

#### Typical usage

Used to carry out an [AM to PM](#) coefficient test.

```
variable = DoAMPMeas(Ptest)
```

#### Action

Performs an AM to PM measurement. Before using this function, an AM to PM calibration must be done, at which time the losses at the input and output of the device under test can be set. A single parameter is passed as described in the table below.

Parameter passed	Description	Value	Unit
Ptest	Power level at which measurement reported	-20 to 0	dBm

#### Returns

Return values	Type	Unit	Executed correctly	Not executed correctly
AM to PM value, "No cal" or "Error"	String	deg/dB	<AM to PM value>	"Error" or "No cal"

## 3.5 Signal processing

### 3.5.1 SetEnhance() - set enhancement parameters

```
Public Function SetEnhance(
    ByVal Para As Variant,
    ByVal k    As Variant
) As String
```

#### Typical usage

Used to set measurement enhancements.

```
variable = SetEnhance(Para, k)
```

#### Action

This function is used to set the various measurement enhancements available. Two parameters are passed ("Para" and "k") as described in the table below.

**Note!** Frequency Sweep must be set using functions such as [SetFreqPlan\(\)](#) and [SelectCal\(\)](#) before this command is used otherwise an error is generated. The set value applies at the next sweep.

Para	Description	Value	Units
"Aver"	Sets number of averages	1 to 255	
"Smoo"	Sets the smoothing	0 to 10	%
"BW"	Sets the measurement bandwidth	10 to 140,000*	Hz
"Powr"	Sets power level	-20 to +6	dBm (0.1 dB steps)
"Dem1"	Sets de-embedding network for port 1	File path and name	
"Dem2"	Sets de-embedding network for port 2	File path and name	
"ApplyDem"	Turns de-embedding on or off	"ON" or "OFF"	
"TDTime"	Sets the start time and stop time for time domain measurements	*See following table	ns
"TDP"	Sets time-domain parameters	*See following table	
"Efd"	Sets the effective dielectric constant of the device under test	1 to 50	
"SynDwell"	Sets extra synthesizer settling time. <b>PicoVNA 108 only.</b>	0 to 255	ms

\*Valid BW settings are: 10, 50, 100, 500, 1000, 5000, 10 000, 15 000, 35 000, 70 000 and 140 000

Para	Argument k	Examples	Notes
"TDTime"	<start time>, <stop time> in ns	"-5, 166" "0, 50"	Start time = -5 ns (min) Stop time = 332 ns (max) Maximum span = 166 ns
"TDP"	<DC termination type>, <resistance value>, <window type>, <window order>	"auto, rect" "res, 50, kaiserb, 6"	DC terminations: "auto", "short", "open", "res". "res" value only needed if termination type is "res". Window types: "rect", "kaiserb", "raisedcos". "window order" value only needed if window type is "kaiserb".

**Returns**

<b>Return values</b>	<b>Type</b>	<b>Executed correctly</b>	<b>Not executed correctly</b>
"OK" or "Error"	String	"OK"	"Error"

### 3.5.2 SetRef() - set reference plane

```
Public Function SetRef(
    ByVal Para As Variant,
    ByVal k As Variant
) As String
```

#### Typical usage

This function is used to set the measurement reference.

```
variable = SetRef(Para, k)
```

#### Action

Used to set the [reference plane](#) of subsequent measurements. A reference plane for each of the four S-parameters can be specified. Two parameters are passed as described in the table below.

Para	Description	k	Units
"S11", "S21", "S12" or "S22"	Measurement parameter to which the reference plane will apply	<value in the range -10 to +10>, "auto" or "?"	metres

The value of k passed to the function must lie within the range -10 to +10 meters. If k is assigned "auto" then the reference plane will automatically be assigned as follows:

Measurement Parameter	"Automatic" reference plane value assigned
S11 or S22	Distance to a short or an open circuit
S21 or S12	Distance required to produce a mean phase of 0°

**Note!** Set the reference plane to 0 before using the "auto" facility.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK", "No Cal" or "Error"	String	"OK"	"Error", "No cal" or "Error"

**Note!** The reference plane is reset to zero after [calibration](#). Therefore, if a value other than zero is required, ensure that it is set after calibration

### 3.5.3 SaveToMem() - save measurement to memory

```
Public Function SaveToMem(
    ByVal Para As Variant
) As String
```

#### Typical usage

Used to stored measured data to memory.

```
variable = SaveToMem(Para)
```

#### Action

This function is used to store measured values to memory. A single parameter is passed as shown in the table below.

Para	Description
"S11" or "S21"	Forward parameters
"S22" or "S12"	Reverse parameters

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

### 3.5.4 AppMemMath() - apply memory math

```
Public Function AppMemMath(
    ByVal Para As Variant,
    ByVal Func As Variant
) As String
```

#### Typical usage

Used to apply vector maths to current measured data.

```
variable = AppMemMath(Para, Func)
```

#### Action

This function is used to apply vector maths to the current measurement only. The parameters passed are described in the table below.

Parameter passed	Description	Values	Comments
Para	Measurement parameter	"S11", "S21", "S12" or "S22"	S22 and S12 are reverse measurements that require the mode to be set to "reverse"
Func	Vector maths to apply	"/", "+", "-", "Restore"	Division, addition and subtraction. "Restore" restores the measurement data to before any math was applied.

**Note!** If this function needs to be called a second time before carrying out another measurement, use the "Restore" facility to restore the measurement data.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

### 3.5.5 SetSysZo() - set system impedance

```
Public Function SetSysZo(
    ByVal SysZo As Integer,
    ByVal ExMatch As Variant
) As String
```

#### Action

This function is used to set the system [impedance](#)  $Z_0$ . The parameters passed as described in the table below.

Parameter passed	Description	Values	Comments
SysZo	System impedance	10 to 200	Only real impedance values
ExMatch	Flag indicating if external impedance matching networks are being used	-	If "True", it is assumed external components are performing the impedance conversion. If "False", the software will perform mathematical conversion from 50 $\Omega$ to the requested value.

**Note:** If external impedance matching networks are being used, a suitable [calibration kit](#) should be used to [calibrate](#). For example, to measure a 75  $\Omega$  device, 75  $\Omega$  to 50  $\Omega$  matching pads could be used and calibration carried out using a 75  $\Omega$  calibration kit. If no impedance matching networks are used, it is assumed that calibration is carried out using a 50  $\Omega$  calibration kit. The software then mathematically converts the measurements to 75  $\Omega$ . Please refer to the *User's Guide* for further information. When applied, data stored in memory is unaffected.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"



### 3.5.6 ZConversion() - impedance conversion utility

```
Public Function ZConversion (
    ByVal ZoOld As Variant,
    ByVal ZoNew As Variant,
    ByVal A11R As Variant,
    A11I As Variant,
    A21R As Double,
    A21I As Variant,
    A12R As Variant,
    A12I As Variant,
    A22R As Double,
    A22I As Variant
) As String
```

#### Action

This function is used to convert normalized S-parameters to a different [impedance](#). The parameters passed are described in the table below.

Parameter passed	Description	Values	Comments
ZoOld	Base impedance of S-parameters to be converted	10 to 200	Only real impedance values
ZoNew	Base impedance of converted S-parameters	10 to 200	Only real impedance values
A11R, A11I	Real and imaginary parts of S11	-	Normalized values
A21R, A21I	Real and imaginary parts of S21	-	Normalized values
A12R, A12I	Real and imaginary parts of S12	-	Normalized values
A22R, A22I	Real and imaginary parts of S22	-	Normalized values

**Note:** The function requires that a full set of s-parameters is passed, i.e. a total of 8 terms representing the real and imaginary parts of each parameter. If a full set is not available, then missing parameters must be given default values. Possible values for any parameter not available are  $10^{-6}$ ,  $j0.0$ . Please be aware that all S-parameters are interrelated so in some circumstances this assumption may not yield best results.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
<string value> or "Error"	String	"S11real, S11imag, S21real, S21imag, S12real, S12imag, S22real, S22imag"	"Error"

## 3.6 Get processed data

### 3.6.1 GetData() - get data

```
Public Function GetData(
    ByVal Para As Variant,
    ByVal MeasType As Variant,
    ByVal Pnt As Variant
) As String
```

#### Typical usage

Used to get measured data (one frequency point per call) in a specified format.

```
variable = GetData(Para, MeasType, Pnt)
```

#### Action

This function is used to get measured data in a specified format. The parameters required to be passed are shown in the table below.

Parameter passed	Description	Values	Comments
Para	Measurement parameter	"S11", "S21", "S12", "S22" or "Rawdata"	"Rawdata" ( <b>PicoVNA 108 only</b> ) only applies after a measurement is done using the "Raw" parameter
MeasType	Required data format	"real", "imag", "logmag", "phase", "swr", "gd", "td"	The available formats are: real, imaginary, log magnitude, phase, standing wave ratio, group delay, time domain
Pnt	Frequency index or time point in <a href="#">CW</a> or TC mode	<frequency index>	Typically a value between 1 and the number of sweep points. 0 or less returns the entire sweep points in comma-separated string

#### Returns

Mode	Return values	Type	Executed correctly	Not executed correctly
Frequency sweep	<string value> or "Error"	String	"<frequency (Hz)>, <parameter value>"	"Error"
CW (time sweep)	<string value> or "Error"	String	"<time (s)>, <parameter value>"	"Error"
TD (time domain)	<string value> or "Error"	String	"<time (s)>, <parameter value>"	"Error"

## 3.6.2 GetMem() - get memory

```
Public Function GetMem(
    ByVal Para      As Variant,
    ByVal MeasType As Variant,
    ByVal Pnt       As Variant
) As String
```

### Typical usage

Used to get memory data (one frequency point per call) in a specified format.

```
variable = GetMem(Para, MeasType, Pnt)
```

### Action

This function is used to get measured data in a specified format. The parameters required to be passed are shown in the table below.

Parameter passed	Description	Values	Comments
Para	Measurement parameter	"S11", "S21", "S12" or "S22"	S22 and S12 are reverse measurements that require the mode to be set to "reverse"
MeasType	Required data format	"real", "imag", "logmag", "phase", "swr", "gd", "td"	The available formats are: real, imaginary, log magnitude, phase, standing wave ratio, group delay, time domain
Pnt	Frequency index	<frequency index>	A value between 1 and the number of sweep points.

### Returns

Return values	Type	Executed correctly	Not executed correctly
<string value> or "Error"	String	"<frequency (Hz)>, <parameter value>"	"Error"

**Note!** Smoothing is not applied to returned memory data.

### 3.6.3 FndPt() - find data point

```
Public Function FndPt(
    ByVal Freq      As Variant,
    ByVal Para      As Variant,
    ByVal MeasType  As Variant,
    ByVal Func      As Variant
) As String
```

#### Typical usage

Gets data in a specified format. Includes facility for finding the peak or minimum value and 3 or 6 dB bandwidth.

```
variable = FndPt(freq, Para, MeasType, Func)
```

#### Action

This function is used to get measured data in a specified format. It can be considered as a marker function. It is similar to [GetData\(\)](#) except that instead of a frequency index, a frequency value (in MHz) is passed. In addition, a parameter can be specified to find the peak or minimum value as well as the 3 or 6 dB bandwidth points. Note that in both cases this will return the measurement points nearest the requested points. The parameters required to be passed are shown in the table below.

Parameter passed	Description	Values	Comments
Freq	Frequency in MHz	"<frequency>"	A value between 0.3 and 6080 (MHz) (PicoVNA 106) or 0.3 and 8500 (PicoVNA 108)
Para	Measurement parameter	"S11", "S21", "S12" or "S22"	S22 and S12 are reverse measurements that require the mode to be set to "reverse"
MeasType	Required data format	"real", "imag", "logmag", "phase", "swr", "gd", "td"	The available formats are: real, imaginary, log magnitude, phase, standing wave ratio, group delay, time domain
Func	"Marker" function required	"normal", "pk", "min", "-3dB", "-6dB", "+3dB", "+6dB"	"Normal" returns the value at the specified frequency. "pk" and "min" find the peak and minimum values, respectively. The other values specify a bandwidth; for example, "-3 dB" refers to the -3 dB bandwidth.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
<string value> or "Error"	String	"<frequency (Hz)>, <(pk, min, etc) value>, <bandwidth value>, <frequency index>"	"Error"

**Note!** Frequency index is an integer representing the point in the sweep.

### 3.6.4 SetLimits() - set pass/fail limits

```
Public Function SetLimits(
    ByVal Segment As Integer,
    ByVal Para As String,
    ByVal FreqLow As Variant,
    ByVal FreqHigh As Variant,
    ByVal Min As Variant,
    ByVal Max As Variant
) As String
```

#### Typical usage

Sets the minimum and maximum pass / fail limits. Up to 8 segments may be specified.

```
variable = SetLimits(Segment, Para, FreqLow, FreqHigh, ValMin, ValMax)
```

#### Action

This function is used to set segments of the pass/fail limits. It is possible to specify up to 8 segments for each measurement parameter (S11, S21, S12 or S22). The parameters required to be passed are described in the table below.

Parameter passed	Description	Values	Comments
Segment	Segment identifier	1 to 8	
Para	Measurement parameter	"S11", "S21", "S12" or "S22"	S22 and S12 are reverse measurements that require the mode to be set to "reverse"
FreqLow	Start frequency of segment in MHz	0.3 to 6080 (or 8500) or "?" or "Reset"	The start frequency of the segment must lie within the <a href="#">calibration</a> frequency band. It must be less than the stop frequency. If FreqLow is set to a string = "?" then the set values of the segment will be returned. Similarly, if it is set to a string = "Reset" the segment is initialized.
FreqHigh	Stop frequency of segment in MHz	0.3 to 6080 or 8500	The stop frequency of the segment must lie within the calibration frequency band. It must be greater than the start frequency.
ValMin	Minimum value limit of segment	-1E9 to 1E9	
ValMax	Maximum value limit of segment	-1E9 to 1E9	

#### Returns

Return values	Type	Executed correctly	Not executed correctly
<string value> or "Error"	String	"OK" or if query, "<start freq (Hz)>, < stop freq (Hz) >, <min value>, <max value>"	"Error"

### 3.6.5 TestLimits() - pass/fail measurement

```
Public Function TestLimits(
    ByVal Para      As String,
    ByVal FreqLow   As Variant,
    ByVal FreqHigh  As Variant,
    ByVal MeasType  As Variant
) As String
```

#### Typical usage

Compares current measurement against the set pass/fail limits. All 8 segments are checked.

```
variable = TestLimits(Para, FreqLow, FreqHigh, MeasType)
```

#### Action

This function is used to test against the pass / fail limits. The parameter (S11, S21, S12 or S22) needs to be specified together with type of measurement. The parameters required to be passed are described in the table below.

Parameter passed	Description	Values	Comments
Para	Measurement parameter	"S11", "S21", "S12" or "S22"	S22 and S12 are reverse measurements that require the mode to be set to "reverse"
FreqLow	Start frequency of test band in MHz	0.3 to (6080 or 8500) or 0	The start frequency of the test band in MHz. If set to 0, then the current <a href="#">calibration</a> 's entire band is used to perform the pass / fail test.
FreqHigh	Stop frequency of test band in MHz	0.3 to (6080 or 8500)	The stop frequency of the test band in MHz.
MeasType	Required data format	"real", "imag", "logmag", "phase", "swr", "gd", "td"	The available formats are: real, imaginary, log magnitude, phase, standing wave ratio, group delay, time domain

#### Returns

Return values	Type	Executed correctly	Not executed correctly
<string value> or "Error"	String	"Pass" or if test fail, "<fail freq (Hz)>, <"Max (or Min) limit =" ValMin (or ValMax)>, <"Actual =">, <Measured value>"	"Error"

## 3.7 Get info

### 3.7.1 GetInfo() - get instrument/cal information

```
Public Function GetInfo(  
    ByVal Para As Variant  
) As String
```

#### Typical usage

This function is used to get data related to the instrument status. A single parameter is passed to the function as described in the table below.

```
variable = GetInfo(Para)
```

Para	Description	Valid return
"ApplyDem"	Flag indicating if de-embedding is on or off	Returns "On" or "Off" to indicate if de-embedding is applied or not
"AverSet"	Number of averages set	1 to 255
"BWSet"	Bandwidth set	10 to 140 000
"CalType"	<a href="#">Calibration</a> type set	"S11+S21", "S11", "S21", "12 Term"
"CalPower"	Signal power used during calibration	-20 to 0 (dBm)
"CalAver"	Averages used in calibration	1 to 16
"CalKit1" or "Calkit2"	Cal kit name loaded for port 1 or port 2	<cal kit name used>
"CWMode"	<a href="#">CW</a> time sweep mode	"False" or "True"
"CFreqPlan"	Queries the <a href="#">Frequency plan</a> used for calibration	Returns a comma-separated string with the frequency plan used. Frequency values returned are in Hz. <start freq>, <stop freq>, <step freq>, <# of sweep points>, <test level>.
"Dem1"	De-embedding network for port 1 loaded-or-not flag	Returns "Loaded" or "Not loaded" if network for port 1 has been loaded or not.
"Dem2"	De-embedding network (port 2) loaded or not	Returns "Loaded" or "Not loaded" to indicate if de-embedding network for port 2 has been loaded
"ErrFlag"	Read serial comms error flag	String. Typically "Error" or "OK"
"ExMatch"	Queries if external <a href="#">impedance</a> matching	Returns a string, "1" if networks used otherwise "0"
"RefP"	<a href="#">Reference plane</a> in use	Value of reference plane in meters and effective dielectric constant
"FreqPlan"	Queries the frequency plan set	Returns a comma-separated string with the current frequency plan. For normal sweep operation frequency values returned are in Hz. <start freq>, <stop freq>, <step freq>, <# of sweep points>, <test level>. In CW time sweep, values returned are <CW freq>, <# of sweep points>, <test level>, <time between pts in ms>.
"Interpol"	Queries if calibration being used is interpolated from another	Returns "Yes" or "No"
"KitSex1" or "KitSex2"	Port 1 or port 2 cal kit sex	"Male" or "Female"

"KitC1" or "KitC2"	Cal kit capacitance coefficients for port 1 kit or port 2 kit	<coefficient 0,coefficient 1, coefficient 2, coefficient 3>
"KitInd1" or "KitInd2"	Cal kit Short circuit inductance for port 1 kit or port 2 kit	<inductance>
"KitOpenOff1" or "KitOpenOff2"	Port 1 or port 2 cal kit Open reference plane	Reference plane of kit in meters and seconds
"KitShortOff1" or "KitShortOff2"	Port 1 or port 2 cal kit Short reference plane	Reference plane of kit in meters and seconds
"KitOpenOffLoss1" or "KitOpenOffLoss2"	Port 1 or port 2 cal kit Open offset loss	Open offset loss in GΩ/s
"KitShortOffLoss1" or "KitShortOffLoss2"	Port 1 or port 2 cal kit Short offset loss	Short offset loss in GΩ/s
"KitL1" or "KitL2"	Port 1 or port 2 cal kit load reflection data flag	Returns 1 if data available or 0 if no data available (considered perfect load)
"KitD1" or "KitD2"	Port 1 or port 2 cal kit load reflection data	Comma-separated string as follows. <# points, frequency (MHz), real part, imaginary part,...>
"Kit1T"	Port 1 cal kit Thru data flag	Returns 1 if data available or 0 if no data available
"Kit1TD"	Port 1 cal kit Thru data (S-parameters)	Comma-separated string as follows. <# points, frequency (MHz), S11r, S11i, S21r, S21i, S12r, S12i, S22r, S22i
"Model"	Instrument model	String. Example: "PicoVNA 106"
"TDP"	Time domain parameters	Returns a comma-separated string with the following data: <Start Time>, <Stop Time>, <Step Time>, <Termination type>,<Res value if applicable>, <Window type>, <Window order (if applicable)>
"SmooSet"	Smoothing set	0 to 10 (%)
"SN"	Instrument serial number	String. Typically a four-digit number.
"HWSN"	Hardware serial number	String. Typically 8 characters long
"Stat"	Instrument status	String: "R": Ready, "S": Ready from power up, "E": Error
"SysZo"	Queries the system impedance set	Returns a string <SysZo>
"Temp"	Instrument temperature (PicoVNA 108 only)	String. <Temp deg C> or "0" if an error is encountered.

### Returns

According to the table above or "Error" if an error has been encountered.



## 3.8 Data storage

### 3.8.1 SaveKit() - save cal kit

```
Public Function SaveKit(
    ByVal FileName As Variant,
    ByVal port      As Integer
) As String
```

#### Typical usage

This function is used to save the [calibration kit](#) on disk. The parameter passed is described in the table below.

```
variable = SaveKit(FileName, port)
```

Parameter passed	Description	Values	Comments
FileName	File name	<path + file name>	The file name must include the full path name
Port	Port to which kit applies	"1" or "2"	

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "File Error"	String	"OK"	"File Error"

**Important:** [Through](#) adaptor data is only loaded and saved when the kit is associated with Port 1 of the instrument. A kit with Thru data may be loaded to Port 2 but the Thru data will be ignored, therefore, if the kit is subsequently resaved the Thru data will be lost.

## 3.8.2 SelectSaveMeas() - save measurement

```
Public Sub SelectSaveMeas()
```

### Typical usage

This routine is called to save measured data. When called, a dialog is displayed that allows the user to select the data and format to be saved as well as the destination file.

No parameters are passed or returned.

### 3.8.3 SaveCal() - save status and calibration

**Note!** [Calibration](#) and status files saved using the remote control [DLL](#) are not compatible with those saved using the PicoVNA 2 or PicoVNA 3 software.

```
Public Function SaveCal(
    ByVal FileName As Variant
) As String
```

#### Typical usage

This function is used to save the current calibration and status. A single parameter is passed as described in the table below.

```
variable = SaveCal(FileName)
```

Parameter passed	Description	Values	Comments
FileName	File name or "?"	<path + file name> or "?"	The file name must include the full path name. If a "?" is passed, a dialog is displayed to allow the user to interactively save the calibration and status.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error writing file"	String	"OK"	"Error writing file"

## 3.9 Data retrieval

### 3.9.1 LoadCal() - load status and calibration

```
Public Function LoadCal(
    ByVal FileName As Variant
) As String
```

#### Typical usage

This function is used to load a [calibration](#) and status file. A single parameter is passed as described in the table below.

```
variable = LoadCal(FileName)
```

Parameter passed	Description	Values	Comments
FileName	File name or "?"	<path + file name> or "<?>"	The file name must include the full path name. If a "?" is passed, a dialog is displayed to allow the user to interactively load a calibration and status file.

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error reading file"	String	"OK"	"Error reading file"

## 3.10 Miscellaneous

### 3.10.1 InitVar() - initialize all variables

Public Function InitVar() as String

#### Typical usage

This function is called to initialize all internal variables.

```
variable = InitVar
```

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK"	String	"OK"	Not applicable

### 3.10.2 SelectSigGen() - set to signal generator mode [dialog]

```
Public Sub SelectSigGen()
```

**Typical usage**

This routine is called to display a dialog that allows the user to set the instrument up as synthesized signal generator.

```
Call SelectSigGen
```

or

```
SelectSigGen
```

**Returns**

No parameters are passed or returned.

### 3.10.3 SetSigGen() - set to signal generator mode

```
Public Function SetSigGen(
    ByVal mode As Variant,
    ByVal P1 As Variant,
    ByVal P2 As Variant,
    ByVal Prt As string) '[PicoVNA 108 only]
) As String
```

#### Typical usage

This function is used to set up the instrument as synthesized signal generator. The parameters passed are described in the table below.

```
variable = SetSigGen("CW", freq, power, "Port1")
```

Mode	Parameter passed	Description	Values	Comments
<a href="#">"CW"</a>	P1	Frequency in MHz	<frequency>	Must be within range 0.3 to (6080 or 8500)
	P2	Power in dBm	<power>	Must be within range: -20 to 6 (PicoVNA 106) -20 to 10 (PicoVNA 108)
	Prt	Output port <b>PicoVNA 108 only</b>	"Port1" or Port2"	

#### Returns

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

**Note:** Setting P2 (power in dBm) to -100 will have the effect of switching off the signal generator output. The maximum output above 6 GHz is +6 dBm.

### 3.10.4 SelectPhaseMeter() - measure phase difference interactively

```
Public Sub SelectPhaseMeter()
```

#### Typical usage

This function is used to measure the phase difference between the two inputs, and their amplitude, interactively. Typically implemented as follows:

```
<string variable> = SelectPhaseMeter()
```

#### Action

A dialog is displayed that allows you to carry out phase measurements interactively. No parameters are passed.

Parameter passed	Description	Value
None		

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"



### 3.10.5 InitialisePhaseMeter() - set frequency and bandwidth

```
Public Sub InitialisePhaseMeter(
    ByVal Fq As Variant,
    ByVal B As Variant
)
```

#### Typical usage

This function sets up parameters for the phase meter. Typically implemented as follows:

```
<string variable> = InitialisePhaseMeter(14, 1000)
```

To take the measurement, call [MeasurePhase\(\)](#).

#### Action

The above initializes the phase meter utility for measurements at 14 MHz with a 1 kHz bandwidth.

Parameter passed	Description	Value
Fq	Frequency for phase measurement (MHz)	0.3 to 8500
B	Receiver bandwidth (Hz)	10 to 140000 in available sequence

Return values	Type	Executed correctly	Not executed correctly
"OK" or "Error"	String	"OK"	"Error"

### 3.10.6 MeasurePhase() - run the phase meter

```
Public Sub MeasurePhase(
    ByVal Mode As String,
    ByVal PhRef As String
)
```

#### Typical usage

This function takes a phase measurement using the phase meter. Frequency and bandwidth must be set first using [InitialisePhaseMeter\(\)](#). Typically implemented as follows:

```
<string variable> = MeasurePhase("mode", "port")
```

#### Action

The above measures phase and amplitude using the specified mode and phase reference.

Parameter passed	Description	Value
Mode	Which correction to apply, if any	"Normalise" – Normalize the phase before taking a measurement "CorrectPortLevel" – Correct the power levels before taking a measurement "Measure" – Only take a measurement
PhRef	Which port to use as the phase reference	"P1" – port 1 "P2" – port 2

Return values	Type	Executed correctly	Not executed correctly
List of results	Comma-separated string	"<phase>, <Port 1 level>, <Port 2 level>, <dB ratio of Port 2 to Port 1 level>"	"Error"

### 3.10.7 SelectSOT() - save on trigger [dialog]

Public Function SelectSOT() As String

#### Typical usage

**PicoVNA 108 only.** This function is used to launch the Save on Trigger form. It is used mainly to examine and / or manipulate captured sweep data.

```
<variable> = SelectSOT()
```

#### Returns

Return values	Type	Executed correctly	Error
"OK" or "Error"	String	"OK"	"Error"

### 3.10.8 SOTMeasure() - do save on trigger measurement

```
Public Function SOTMeasure(
    ByVal Swps As variant,
    ByVal Trg As string,
    ByVal para As string,
    ByVal Tout As Variant
) As String
```

**Typical usage**

**PicoVNA 108 only.** This function is used to carry out measurement sweeps with Save on Trigger enabled using the current calibration. It will store all measured data so that it can be inspected later using the [SelectSOT\(\)](#) command or downloaded to the user application with the [GetSOTData\(\)](#) command.

```
<variable> = SOTMeasure(16, "pos")
```

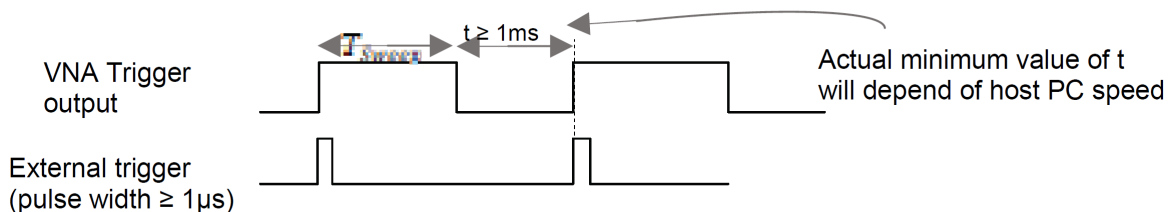
Parameter passed	Description	Values	Comments
Swps	Number of sweeps to capture	<2 to 1024>	Number of measurement sweeps to capture
Trg	Trigger mode	"Pos" or "Neg"	Select positive or negative edge external trigger mode. Any other value will return an error.
Para	Parameter	"S11", "S21", "S11+S21", "All"	Parameter to be measured
Tout	Time out	<1 to 1200000>	Timeout in milliseconds if no trigger is received (1 ms to 20 minutes)

**Returns**

Return values	Type	Executed correctly	Error
"2" to "1024" or "Error"	String	"<number of sweeps captured>"	"Error"

**External trigger considerations**

Carefully consider the maximum possible external trigger rate. When using the DLL to control the instrument, a faster trigger rate than when using the UI software is possible. This is because the host PC does not need to deal with the graphics display. So, for example, with a 201-point sweep plan and using the maximum bandwidth of 140 kHz it should be possible to use external trigger rates of 20 Hz or more without any missing any trigger events. The diagram below shows the trigger timing diagram.  $T_{sweep}$  depends on the number of points in the sweep plan and bandwidth setting. The user can check this by observing the trigger output from the instrument with the trigger set to free run.



**Note:** The time out value parameter determines when the command is aborted in cases when no external trigger is received. This value is in addition to the built-in default time out which is dependent on parameters such as the bandwidth setting.

### 3.10.9 SOT\_CopySweep() - copy to trace

```
Public Function SOT_CopySweep(
    ByVal Swp As Integer,
    ByVal Trc As String
) As String
```

#### Typical usage

**PicoVNA 108 only.** This function is used to copy a saved sweep to the current live data array or memory data array. Thereafter the data can be retrieved using [GetData\(\)](#) for example.

```
<variable> = SOT_CopySweep(16, "Live")
```

Parameter passed	Description	Values	Comments
Swp	Saved sweep to copy	<1 to 1024>	Captured measurement sweep to copy
Trc	Trace array to receive data	<"Live" or "Memory">	

#### Returns

Return values	Type	Executed correctly	Error
"OK" or error description	String	"OK"	"Error" or "Invalid trace name", etc.

### 3.10.10 SOT\_SaveData() - save sweep to data array

```
Public Function SOT_CopySweep(
    ByVal File As String,
    ByVal Para As String,
    ByVal Format As String,
    ByVal Mode As String,
    ByVal Freq As Variant
) As String
```

#### Typical usage

**PicoVNA 108 only.** This function is used to copy a saved sweep to the current live data array or memory data array. Thereafter the data can be retrieved using [GetData\(\)](#) for example.

```
<variable> = SOT_SaveData("C:\Temp\Mydata.s2p", "S11,S21,S12,S22", "Touch",
    "SortBySweep", 0)
```

Parameter passed	Description	Values	Comments
File	Disk file	<file name>	
Para	S-parameter to be saved	<"S11" or "S11,S21" etc>	Comma-separated list of s-parameter names with no spaces
Format	Data format to be used	<"LogMag" or "Mag" or "Real" or "Touch" or "logMag_only" or "Phase" or "Mag_only">	Saved data format
Mode	Mode to use for saving the data to disk	<"SortByPara" or "SortByPara1" or "SortBySweep" or "SortBySweep1">	Sort data by S-parameter or by sweep. Adding "1" to the name saves the data at only one frequency.
Freq	Frequency value	<frequency value>	Applies only when saving data for a single frequency.

#### Returns

Return values	Type	Executed correctly	Error
"OK" or error description	String	"OK"	"Error" or "Invalid file"

### 3.10.11 DLLVer() - get DLL program version

Public Function DLLVer() As String

#### Typical usage

This function is used to query the [DLL](#) program version. No parameters are passed.

```
variable = DLLVer
```

#### Returns

Return values	Type	Executed correctly	Comments
<version>	String	<version>	Example: "V1.0 Jan 2014"

## 3.11 Diagnostics

### 3.11.1 InsDiag() - run diagnostics tests [dialog]

Public Function InsDiag() As String

#### Typical usage

This function is used to display a dialog that allows the user to carry out instrument diagnostics tests. No parameters are passed.

```
variable = InsDiag
```

#### Returns

Return values	Type	Executed correctly	Not executed (aborted)
<string>	String	"Tests completed"	"Tests not performed"



## 4 Glossary

**AM to PM.** Amplitude modulation to phase modulation conversion. An undesirable effect in which changes in amplitude in a system cause changes in phase. This can cause data errors in a digital system.

**Calibration.** A procedure for testing the VNA at regular intervals to measure its inherent errors. These errors are then subtracted from the measurements made on the DUT to obtain a more accurate result.

**Calibration kit.** To remove errors caused by impedance mismatches and nonlinearities in the VNA and its associated cables and connectors, it must be calibrated at frequent intervals. A calibration kit, consisting of some combination of short circuit, open circuit, known load, and through-adaptors, is required for this process. The calibration kit has known electrical parameters that must be stored in the VNA software.

**CW.** Continuous wave. A fixed-amplitude sine wave.

**DLL.** Dynamic link library. A collection of functions that can be called to control the VNA and retrieve data from it. The functions can be called from Microsoft VBA and other compatible programming languages.

**DUT.** Device under test. The component or system connected through cables and adaptors to the VNA's test ports.

**EEPROM.** Electrically erasable read-only memory. A memory device inside the VNA that stores configuration information such as the device's serial number.

**Frequency plan.** A list of frequencies and power levels that the VNA will generate during the test.

**Impedance.** The system impedance ( $Z_0$ ) is assumed to match that of the VNA, which is 50  $\Omega$ . If it does not, the VNA can compensate for the actual system impedance.

**P1dB.** 1 dB gain compression point. An ideal DUT has constant gain regardless of the power level applied to it. In practice, the gain tends to decrease at higher power levels. P1dB is the input power level at which the gain drops by 1 dB from its nominal value.

**Sweep trigger.** The frequency sweep generated by the VNA can be free-running or synchronized to an external signal. In the latter case, the frequency sweep starts when the VNA is triggered by the external signal.

**Reference plane.** A location somewhere between the VNA output and the DUT input used as the zero reference for measuring distance. It can be adjusted by the software to account for the effect of adding through adaptors.

**Through.** An adaptor that does not intentionally modify the signal passing through it but functions only to adapt the connector on one device to fit another device.

# Index

## A

- AM to PM 65
- AM to PM conversion
  - calibration 27
  - dialog 25
  - measuring 34
- AppMemMath() 39

## C

- Calibrating 20
- Calibration 65
- Calibration and status
  - loading 52
  - saving 51
- Calibration kit 65
  - creating 16
  - creating or loading 15
  - saving 49
- Calibration standard, measuring 21
- CloseVNA() 13
- Closing an instrument 13
- Contact Pico 6
- Continuous wave 65
- Conversion
  - amplitude modulation to phase modulation 65
- CPU 6
- CW 65
- CW sweep plan, setting 19

## D

- Data point, finding 44
- Data, requesting 42
- Device under test 65
- Diagnostics tests, running 64
- Discover instrument 11
  - by serial number 12
- Disk space 6
- DLL 65
- DLL version, requesting 63
- DLLVer() 63
- DoAMPMeas() 27
- DoAMPMeas() 34
- DoP1dBCal() 26
- DoP1dBMeas() 33
- DUT 65
- Dynamic link library 65

## E

- EEPROM 65
- Enhancement parameters, setting 35

## F

- FndPt() 44
- FNDSN() 12
- Frequency plan 65
- Frequency plan, setting 18, 20

## G

- GetData() 42
- GetInfo() 47
- GetMem() 43

## I

- Impedance 65
- Impedance conversion utility 41
- InitialisePhaseMeter() 57
- Initializing variables 53
- InitVar() 53
- InsDiag() 64
- Installation 7
- Instrument status 47

## L

- License conditions 4
- LoadCal() 52

## M

- MeasCal() 21
- Measure() 32
- Measurement, saving 50
- MeasurePhase() 58
- Memory math, applying 39
- Memory, requesting 43
- Microsoft Windows 6
- Mixer
  - selecting 28

## O

- Operating system 6

## P

- P1dB 65

- P1dB 65
    - calibrating 26
    - measuring 24, 33
  - Pass/fail limits, setting 45
  - Pass/fail measurement 46
  - Phase meter
    - initializing 57
    - interactive 56
    - running 58
  - PicoVNA 4
  - Processor 6
  - PulseTrigOut() 31
- ## Q
- Quiescent point, setting 29
- ## R
- Reference plane 65
  - Reference plane, setting 37
  - ResetErr() 14
- ## S
- Sales support 6
  - Save measurement to memory 38
  - Save on trigger
    - copy to live or memory 61
    - measuring 60
    - save data 62
    - selecting 59
  - SaveCal() 51
  - SaveKit() 49
  - SaveToMem() 38
  - SelectCal() 20
  - SelectKit() 15
  - SelectMixerFrm() 28
  - SelectP1dB() 24
  - SelectPhaseMeter() 56
  - SelectSaveMeas() 50
  - SelectSigGen() 54
  - SelectSOT() 59
  - Serial communications error flag, resetting 14
  - SetCWmode() 19
  - SetEnhance() 35
  - SetFreqPlan() 18
  - SetKit() 16
  - SetLimits() 45
  - SetRef() 37
  - SetRXQ() 29
  - SetSigGen() 55
  - SetSysZo() 40
  - SetTrig() 30
  - Signal generator mode, setting 54, 55
  - SOT\_CopySweep() 61
  - SOT\_SaveData() 62
  - SOTMeasure() 60
  - Status and calibration
    - loading 52
    - saving 51
  - Sweep trigger 65
  - Sweep trigger mode, setting 30
  - Sweep, measuring one 32
  - System impedance, setting 40
  - System requirements 6
- ## T
- Technical support 6
  - TestLimits() 46
  - Through 65
  - Trademarks 5
  - Trigger output, pulsing 31
- ## U
- USB 6
- ## V
- Variables, initializing 53
- ## Z
- ZConversion() 41

**United Kingdom global headquarters:**  
Pico Technology  
James House  
Colmworth Business Park  
St. Neots  
Cambridgeshire  
PE19 8YP  
United Kingdom

Tel: +44 (0) 1480 396 395

[sales@picotech.com](mailto:sales@picotech.com)  
[support@picotech.com](mailto:support@picotech.com)

[www.picotech.com](http://www.picotech.com)

**North America regional office:**  
Pico Technology  
320 N Glenwood Blvd  
Tyler  
TX 75702  
United States

Tel: +1 800 591 2796

[sales@picotech.com](mailto:sales@picotech.com)  
[support@picotech.com](mailto:support@picotech.com)

**Asia-Pacific regional office:**  
Pico Technology  
Room 2252, 22/F, Centro  
568 Hengfeng Road  
Zhabei District  
Shanghai 200070  
PR China

Tel: +86 21 2226-5152

[pico.asia-pacific@picotech.com](mailto:pico.asia-pacific@picotech.com)