NAME

vnacal_new_alloc, vnacal_new_free, vnacal_new_set_frequency_vector, vnacal_new_set_z0, vnacal_new_add_single_reflect, vnacal_new_add_single_reflect_m, vnacal_new_add_double_reflect, vnacal_new_add_double_reflect_m, vnacal_new_add_through, vnacal_new_add_through_m, vnacal_new_add_line, vnacal_new_add_line_m, vnacal_new_add_mapped_matrix, vnacal_new_add_mapped_matrix_m, vnacal_new_solve, vnacal_new_set_m_error, vnacal_new_set_et_tolerance, vnacal_new_set_p_tolerance, vnacal_new_set_iteration_limit, vnacal_new_set_pvalue_limit – find error terms from measured standards

SYNOPSIS

#include <vnacal.h>

Starting a New Calibration

vnacal_new_t *vnacal_new_alloc(vnacal_t ***_vcp_, vnacal_type_t** _type_**,**
int _rows_**, int** _columns_**, int** _frequencies_**);**

int vnacal_new_set_frequency_vector(vnacal_new_t ***_vnp_, const double ***_frequency_vector_**);**

int vnacal_new_set_z0(vnacal_new_t ***_vnp_, double complex** _z0_**);**

Adding Measured Standards

int vnacal_new_add_single_reflect(vnacal_new_t ***_vnp_,**
double complex *const ***_a_, int** _a_rows_**, int** _a_columns_**,**
double complex *const ***_b_, int** _b_rows_**, int** _b_columns_**,**
int _s11_**, int** _port_**);**

int vnacal_new_add_single_reflect_m(vnacal_new_t ***_vnp_,**
double complex *const ***_m_, int** _m_rows_**, int** _m_columns_**,**
int _s11_**, int** _port_**);**

int vnacal_new_add_double_reflect(vnacal_new_t ***_vnp_,**
double complex *const ***_a_, int** _a_rows_**, int** _a_columns_**,**
double complex *const ***_b_, int** _b_rows_**, int** _b_columns_**,**
int _s11_**, int** _s22_**, int** _port1_**, int** _port2_**);**

int vnacal_new_add_double_reflect_m(vnacal_new_t ***_vnp_,**
double complex *const ***_m_, int** _m_rows_**, int** _m_columns_**,**
int _s11_**, int** _s22_**, int** _port1_**, int** _port2_**);**

int vnacal_new_add_through(vnacal_new_t ***_vnp_,**
double complex *const ***_a_, int** _a_rows_**, int** _a_columns_**,**
double complex *const ***_b_, int** _b_rows_**, int** _b_columns_**,**
int _port1_**, int** _port2_**);**

int vnacal_new_add_through_m(vnacal_new_t ***_vnp_,**
double complex *const ***_m_, int** _m_rows_**, int** _m_columns_**,**
int _port1_**, int** _port2_**);**

int vnacal_new_add_line(vnacal_new_t ***_vnp_,**
double complex *const ***_a_, int** _a_rows_**, int** _a_columns_**,**
double complex *const ***_b_, int** _b_rows_**, int** _b_columns_**,**
const int ***_s_2x2_, int** _port1_**, int** _port2_**);**

int vnacal_new_add_line_m(vnacal_new_t ***_vnp_,**
double complex *const ***_m_, int** _m_rows_**, int** _m_columns_**,**
const int ***_s_2x2_, int** _port1_**, int** _port2_**);**

int vnacal_new_add_mapped_matrix(vnacal_new_t ***_vnp_,**
double complex *const ***_a_, int** _a_rows_**, int** _a_columns_**,**
double complex *const ***_b_, int** _b_rows_**, int** _b_columns_**,**
const int ***_s_, int** _s_rows_**, int** _s_columns_**,**
const int ***_port_map_);**

int vnacal_new_add_mapped_matrix_m(vnacal_new_t **vnp*, **double complex *const** **m*, **int** *m_rows*, **int** *m_columns*, **const int** **s*, **int** *s_rows*, **int** *s_columns*, **const int** **port_map*);

Solving for the Error Terms

int vnacal_new_solve(vnacal_new_t **vnp*);

Cleanup

void vnacal_new_free(vnacal_new_t **vnp*);

Managing Measurement Error and Tolerance

int vnacal_new_set_m_error(vnacal_new_t **vnp*, **const double** **frequency_vector*, **int** *frequencies*, **const double** **sigma_nf_vector*, **const double** **sigma_tr_vector*);

int vnacal_new_set_pvalue_limit(vnacal_new_t **vnp*, **double** *significance*);

int vnacal_new_set_et_tolerance(vnacal_new_t **vnp*, **double** *tolerance*);

int vnacal_new_set_p_tolerance(vnacal_new_t **vnp*, **double** *tolerance*);

int vnacal_new_set_iteration_limit(vnacal_new_t **vnp*, **int** *iterations*);

Link with *-lvna -lyaml -lm*.

DESCRIPTION

These functions find error terms for vector network analyzers (VNAs) from measurements of calibration standards. The resulting error terms can be subsequently used to correct for errors when measuring an unknown device under test (DUT).

The overall flow is as follows. Begin by calling **vnacal_create**() or **vnacal_load**() to obtain a pointer to a **vnacal_t** structure. See **vnacal**(3). Next, start a new calibration by calling **vnacal_new_alloc**(), giving the error term type, dimensions of the calibration, and the number of frequency points to be used in the calibration. Load the calibration frequencies using **vnacal_new_set_frequency_vector**(), and set the system impedance using **vnacal_new_set_z0**(). Next, make measurements of calibration standards and add the results of each using the various **vnacal_new_add_***() functions. Solve for the error terms using **vnacal_new_solve**(). Finally, add the resulting new calibration to the **vnacal_t** structure using **vnacal_add_calibration**().

The library supports several types of error terms, capable of correcting different errors. In general, error term types with more terms correct for more errors, but at a cost of requiring more measured standards. The *type* parameter to **vnacal_new_alloc**() can be any of the following values:

VNACAL_T8	8-term T terms
VNACAL_U8	8-term U (inverse T) terms
VNACAL_TE10	8-term T and off-diagonal leakage terms
VNACAL_UE10	8-term U and off-diagonal leakage terms
VNACAL_T16	16-term T terms
VNACAL_U16	16-term U (inverse T) terms
VNACAL_UE14	14-term columns x (rows x 1) U7 systems
VNACAL_E12	12-term generalized classic SOLT

Error term types are covered in more detail below.

The *rows* and *columns* parameters give the dimensions of the calibration, where *rows* is the number of VNA ports that detect signal and *columns* is the number of VNA ports that transmit signal. Usually, all VNA ports can both transmit and detect signal, and both *rows* and *columns* are simply the number of VNA ports. But some vector network analyzers, such as certain models of the N2PK open source hardware VNA, transmit and detect signal on port 1 as usual, but only detect signal on port 2. For this VNA, *rows* is 2 and *columns* is 1. If instead of having detect-only ports, a given VNA has some number of transmit-only ports, it can be modeled by making *columns* greater than *rows*. In all cases, however, the low-numbered ports forming the square portion of the matrix must be able to both transmit and detect, i.e. for the purpose of calibration, any detect-only or transmit-only ports must be given the highest port numbers, and the VNA cannot have both.

Error Term Types

The following table illustrates some properties of the various error term types. First, there are constraints on the type imposed by the calibration dimensions. If the calibration has more columns than rows, T parameters must be used; if it has more rows than columns, U or E12 parameters must be used. Next, the actual number of error terms depends on the calibration dimensions, shown for both the square case (*rows* = *columns* = ports) and in the general rectangular case. When the dimensions are 2x2, the number of error terms matches the number in the calibration type name. Finally, all T and U types contain an arbitrary term. For example, in 2x2 T8, one of the eight terms is a free variable, so we have to solve for only seven unknowns. Similarly, in 2x2 T16, one variable is free and we have to solve for only fifteen unknowns. In the table, r is rows, c is columns and p is ports.

type	**constraints**	**error terms**		**free**
		square	**rectangular**	
T8	r <= c	4p	2r + 2c	1
U8	r >= c	4p	2r + 2c	1
TE10	r <= c	p^2 + 3p	rc + r + 2c	1
UE10	r >= c	p^2 + 3p	rc + 2r + c	1
T16	r <= c	4p^2	2rc + 2c^2	1
U16	r >= c	4p^2	2rc + 2r^2	1
UE14	r >= c	3p^2 + p	3rc + c	c
E12	r >= c	3p^2	3rc	0

T8 & U8

These types correct for directivity, reflection / transmission tracking, and port match errors on each VNA port. Notice from the table that these are the only types for which the number of error terms increases proportionally to the number of VNA ports, i.e. this correction has no inter-port terms. At least three standards are needed to solve the 2x2 T8 or U8 calibration.

TE10 & UE10

These types correct the same errors as T8 & U8 but also add the off-diagonal leakage terms, i.e. leakage within the VNA from the driving port to the detectors of the other ports. At least three standards are needed to solve the 2x2 TE10 or UE10 calibration.

T16 & U16

These types are a superset of TE10 & UE10, adding the remaining leakage terms, including leakage between the ports of the device under test. At least five standards are needed to solve the 2x2 T16 or U16 calibration.

UE14

This type corrects the same errors as TE10 and UE10, but is stronger in that it treats each column (driving port) as an independent calibration, i.e. it's a columns long series of rows x 1 independent systems. Because of this, this type is able to correct for errors in switches, even in a switch that lies between the detectors and the device under test. At least four standards are needed to solve the 2x2 UE14 calibration.

E12

E12 is a generalization of classic SOLT. Internally, library solves the system using UE14 terms and thus corrects for exactly the same errors as EU14, creating separate calibrations for each column. The difference is only in the format of the saved error terms. In E12, the library converts the U error terms it used to solve the systems to the more conventional E (scattering parameter) form before saving them. At least four standards are needed to solve the 2x2 E12 calibration.

S-Parameters of the Standards

All **vnacal_new_add_***() functions except for **vnacal_new_add_through**() take one or more S-parameters describing the calibration standard. Instead of taking complex values for the S-parameters directly, these functions take integer values that can be either one of the predefined constants: **VNACAL_MATCH**, **VNACAL_OPEN**, **VNACAL_SHORT**, **VNACAL_ZERO**, **VNACAL_ONE**; or an integer handle returned from one of the **vnacal_make_*_parameter**() functions. See **vnacal_parameter**(3). There are two main reasons for this approach. First, it provides a single interface for parameters that are constant across all frequencies (e.g. -1.0 for short), and parameters that are given at a list of frequency points. Second, it allows for parameters to be specified as unknown – parameters that the library has to solve for.

Measurements

The **vnacal_new_add_***() functions come in pairs with one taking separate *a* and *b* matrices, and the other, a single *m* matrix. If the VNA measures both the voltage leaving each port (a), and the voltage entering each port (b), use the *a,b* form. This form gives more accurate results because it corrects for variations in signal generator output level and for errors in a switch that lies between the signal generator and directional couplers. If the VNA doesn't measure separate a and b parameters, then the *m* form can be used. Always use the same form for error correction as was used for calibration.

For T8, U8, TE10, UE10, T16 and U16 error term types, the *a* matrix has dimensions *b_columns* x *b_columns*. The rows of *a* represent the amount of signal leaving the respective VNA port; the columns of *a* represent the VNA port that was nominally driving signal when the values in the column were measured. When *a* and *b* matrices are given with these error term types, the library calculates the measurement matrix using ab^{-1}. For E12 and UE14 error terms, the calibration is a *columns* long sequence of independent *rows* x 1 systems; therefore, *a* is a row of 1x1 matrices, or equivalently a row vector of reference values. Because each column is a separate system, these calibration types correct for errors in a switch, even a switch that lies between the directional couplers and DUT.

It's always permitted to specify the full *rows* x *columns* measurement matrix representing all VNA ports, even if the associated calibration standard has fewer ports. This is useful for determining leakage terms in calibration types that correct for them. In most cases, it's also possible to give an abbreviated measurement matrix in which the number of rows or the number of columns is equal to the number of ports of the standard being measured. For example, in type T8, if we're adding a reflect standard on a single VNA port, the measurement matrix can be *rows* x *columns*, 1 x *columns*, *rows* x 1, or 1x1. In T16, however, *b_columns* or *m_columns* must be the full set of calibration columns, and in U16, *b_rows* or *m_rows* must be the full set of calibration rows.

When an abbreviated measurement matrix is given, the abbreviated rows or columns always appear in port number order, even if the ports of the standard are mapped out of order. For example, if the VNA has four transmit/detect ports (*rows* = 4 and *columns = 4*), and we're adding a 2x2 measurement matrix for a short-open double reflect standard with *port1*=3 and *port2*=2, the first element of the measurement matrix represents the open on VNA port 2.

If the standard has fewer ports than the VNA, the S-parameters measured by the unused VNA ports don't matter as long as they remain constant over the measurement, and as long as, except for leakage, they have no through signal to or from the ports under test. When possible, terminate unused VNA ports with loads close to the system impedance to avoid adding unnecessary noise into the leakage measurements.

Starting a New Calibration

vnacal_new_alloc() creates a structure of type **vnacal_new_t** used by all of the other functions. The *vcp* parameter is a pointer to a **vnacal_t** structure obtained from **vnacal_create**() or **vnacal_load**(). The *type*, *rows* and *columns* parameters determine the type and dimensions of error terms as described above. The *frequencies* parameter gives the number of frequency points in the calibration.

vnacal_new_set_frequency_vector() copies a vector of calibration frequency points into the **vnacal_new_t** structure; *frequency_vector* must point to a vector of non-negative and ascending values with length equal to the *frequencies* argument given to **vnacal_new_alloc**().

vnacal_new_set_z0() sets the system impedance for the vector network analyzer. If not called, the value defaults to 50 ohms. The library assumes all VNA ports have the same reference impedance.

Adding Measured Standards

vnacal_new_add_single_reflect() and **vnacal_new_add_single_reflect_m**() add the measurement of a single port standard with parameter handle *s11* on VNA port *port*. See **vnacal_parameter**(3).

vnacal_new_add_double_reflect() and **vnacal_new_add_double_reflect_m**() add the measurement of two reflect standards with parameter handles *s11* and *s22*, on VNA ports *port1* and *port2*, respectively. The s12 and s21 parameters of the standard must be zero.

vnacal_new_add_through() and **vnacal_new_add_through_m**() add the measurement of a perfect through standard (s11 = 0, s12 = 1, s21 = 1, s22 = 0) between VNA ports *port1* and *port2*.

vnacal_new_add_line() and **vnacal_new_add_line_m**() add the measurement of an arbitrary 2x2 standard on VNA ports *port1* and *port2*. The *s_2x2* argument is a pointer to the first element of a 2x2 matrix of parameter handles.

vnacal_new_add_mapped_matrix() and **vnacal_new_add_mapped_matrix_m**() add the measurement of an arbitrary multi-port standard. The *s* parameter is a pointer to the first element of an *s_rows* x *s_columns* matrix of s-parameter handles. The *port_map* parameter is a vector of length max(*s_rows*, *s_columns*), one entry for each DUT port, containing VNA port numbers, describing which VNA ports are connected to the respective ports of the standard. It may be NULL if the number of VNA ports is equal to the number of ports of the standard and the ports are connected in order.

Solving for the Error Terms

vnacal_new_solve() uses the added measurements to solve for the error terms. It returns 0 on success and -1 on error. After calling **vnacal_new_solve**(), it is permitted to add additional measurements and repeat the call, for example if **vnacal_new_solve**() fails due to an insufficient number of standards.

Cleanup

vnacal_new_free() frees the **vnacal_t** structure, the contained frequency vector, added measurements and error terms. Note that a call to **vnacal_free**() implicitly frees all associated **vnacal_t** structures; don't call **vnacal_new_free**() after calling **vnacal_free**().

Managing Measurement Error and Tolerance

vnacal_new_set_m_error() enables measurement error modeling in **vnacal_new_solve**(). Specifying the measurement errors with this function can significantly reduce error in the solved error terms, especially when the system of measured standards is significantly overdetermined, as is usually the case when using the 16 error term models. The *sigma_nf_vector* parameter is a *frequencies* long vector of standard deviations of noise floor measurements at the VNA detectors when no signal is applied. Similarly, *sigma_tr_vector* is *frequencies* long vector of standard deviations describing an additional noise source, e.g. random amplitude modulation in the signal generator, that is proportional to the RMS amplitude of the received signal. The later is optional and may be given as NULL. Both noise sources are assumed to be Gaussian and i.i.d. If both vectors are given as NULL, then measurement error modeling is reset to disabled.

Note that when separate *a* and *b* measurements are given, the amplitude used for scaling *sigma_tr_vector* is that after dividing the *b* matrix by the *a* matrix. If the *a* matrix and *b* matrix are measured simultaneously, then amplitude modulation error in the signal source will have already been divided out, and *sigma_tr_vector* is likely not useful.

The *frequency_vector* parameter is a vector of ascending frequency values where the noise measurements were made. These frequencies don't have to align with the frequencies given in **vnacal_new_set_frequency_vector**() – the library uses cubic spline interpolation as necessary – but the frequency range must span the entire range of calibration frequencies. The **vnacal_new_set_frequency_vector**() function must be called before **vnacal_new_set_m_error**(). If *frequencies* is 1, then *frequency_vector* is not used and can be specified as NULL. In this case, the single noise values given apply to all frequencies. If *frequencies* is equal to the number of frequencies given in **vnacal_new_set_frequency_vector**() and *frequency_vector* is NULL, then the frequency vector defaults to that given in **vnacal_new_set_frequency_vector**().

When using VNACAL_T16 or VNACAL_U16 error term types with measurement error modeling, the complete s-parameter matrix for each calibration standard must be given; when not using measurement error modeling, not all cells of these matrices are required to be known.

When measurement error modeling is enabled, **vnacal_new_solve**() computes a p-value giving the probability that the magnitudes of the residuals observed are less than or equal to those expected due to random errors, assuming that the measurements are consistent with the specifed error model. The **vnacal_new_set_pvalue_limit**() function sets the *significance* below which the library should reject the null hypothesis that the measurements are consistent. In this case, **vnacal_new_solve**() returns failure. *significance* must be greater than zero and no more than one. The default is 0.001.

Some of the solve methods used in **vnacal_new_solve**() are iterative and some are analytical. If we're modeling measurement errors (see **vnacal_new_set_m_error**()), the solution is always iterative. Here, the library weights the equations such that the residuals reflect the expected measurement errors. The solved error terms depend on the weights, but the weights depend on the solved error terms, thus the library iterates until these converge. When there are unknown parameters in the calibration standards, the solution method is usually iterative, except in the special case of two-port TRL, which has an analytical solution. The following functions control the iterative methods.

vnacal_new_set_et_tolerance() sets the degree of RMS change in the error terms sufficiently small to stop iteration. Similarly **vnacal_new_set_p_tolerance**() sets the degree of RMS change in the unknown parameters sufficiently small to stop iteration. Both tolerances must be met before the system is considered to be converged. The default value for both functions is 1.0e-6.

vnacal_new_set_iteration_limit() sets the maximum number of iterations allowed for convergence. If the system has still not converged by this limit, then **vnacal_new_solve**() fails. The default is 30.

RETURN VALUE

The **vnacal_new_alloc**() function returns a pointer to an opaque **vnacal_new_t** structure needed by the other functions. All integer-valued functions return 0 on success or -1 on error.

ERRORS

On error, these functions invoke the *error_fn*, given to **vnacal_create**() or **vnacal_load**() if not NULL, set **errno** to one of the following values and return failure.

EDOM Too few measured standards were given, the system is singular or the solution did not converge.

ENOMEM
The library was unable to allocate memory.

EINVAL A function was called with an invalid parameter.

SEE ALSO

vnacal(3), **vnaconv**(3), **vnadata**(3), **vnaerr**(3), **vnacal_parameter**(3)