NAME

vnaconv_atob, vnaconv_atog, vnaconv_atoh, vnaconv_atos, vnaconv_atot, vnaconv_atou, vnaconv_atoy, vnaconv_atoz, vnaconv_atozi, vnaconv_btoa, vnaconv_btog, vnaconv_btoh, vnaconv_btos, vnaconv_btot, vnaconv_btou, vnaconv_btoy, vnaconv_btoz, vnaconv_btozi, vnaconv_gtoa, vnaconv_gtob, vnaconv_gtoh, vnaconv_gtos, vnaconv_gtot, vnaconv_gtou, vnaconv_gtoy, vnaconv_gtoz, vnaconv_gtozi, vnaconv_htoa, vnaconv_htob, vnaconv_htog, vnaconv_htos, vnaconv_htot, vnaconv_htou, vnaconv_htoy, vnaconv_htoz, vnaconv_htozi, vnaconv_stoa, vnaconv_stob, vnaconv_stog, vnaconv_stoh, vnaconv_stot, vnaconv_stou, vnaconv_stoy, vnaconv_stoyn, vnaconv_stoz, vnaconv_stozi, vnaconv_stozin, vnaconv_stozn, vnaconv_ttoa, vnaconv_ttob, vnaconv_ttog, vnaconv_ttoh, vnaconv_ttos, vnaconv_ttou, vnaconv_ttoy, vnaconv_ttoz, vnaconv_ttozi, vnaconv_utoa, vnaconv_utob, vnaconv_utog, vnaconv_utoh, vnaconv_utos, vnaconv_utot, vnaconv_utoy, vnaconv_utoz, vnaconv_utozi, vnaconv_ytoa, vnaconv_ytob, vnaconv_ytog, vnaconv_ytoh, vnaconv_ytos, vnaconv_ytosn, vnaconv_ytot, vnaconv_ytou, vnaconv_ytoz, vnaconv_ytozi, vnaconv_ytozin, vnaconv_ytozn, vnaconv_ztoa, vnaconv_ztob, vnaconv_ztog, vnaconv_ztoh, vnaconv_ztos, vnaconv_ztosn, vnaconv_ztot, vnaconv_ztou, vnaconv_ztoy, vnaconv_ztoyn, vnaconv_ztozi, vnaconv_ztozin – convert 2-port and N-port network parameters

SYNOPSIS

Include and Link

#include <vnaconv.h>

Link with *-lvna -lm*.

Two-Port Matrix Conversions

void vnaconv_stot(const double complex *s***[2][2], double complex** *t***[2][2]);**

void vnaconv_stou(const double complex *s***[2][2], double complex** *u***[2][2]);**

void vnaconv_stoz(const double complex *s***[2][2], double complex** *z***[2][2],
const double complex** *z0***[2]);**

void vnaconv_stoy(const double complex *s***[2][2], double complex** *y***[2][2],
const double complex** *z0***[2]);**

void vnaconv_stoh(const double complex *s***[2][2], double complex** *h***[2][2],
const double complex** *z0***[2]);**

void vnaconv_stog(const double complex *s***[2][2], double complex** *g***[2][2],
const double complex** *z0***[2]);**

void vnaconv_stoa(const double complex *s***[2][2], double complex** *a***[2][2],
const double complex** *z0***[2]);**

void vnaconv_stob(const double complex *s***[2][2], double complex** *b***[2][2],
const double complex** *z0***[2]);**

void vnaconv_ttos(const double complex *t***[2][2], double complex** *s***[2][2]);**

void vnaconv_ttou(const double complex *t***[2][2], double complex** *u***[2][2]);**

void vnaconv_ttoz(const double complex *t***[2][2], double complex** *z***[2][2],
const double complex** *z0***[2]);**

void vnaconv_ttoy(const double complex *t***[2][2], double complex** *y***[2][2],
const double complex** *z0***[2]);**

void vnaconv_ttoh(const double complex *t***[2][2], double complex** *h***[2][2],
const double complex** *z0***[2]);**

void vnaconv_ttog(const double complex *t***[2][2], double complex** *g***[2][2],
const double complex** *z0***[2]);**

void vnaconv_ttoa(const double complex *t***[2][2], double complex** *a***[2][2],
const double complex** *z0***[2]);**

void vnaconv_ttob(const double complex *t***[2][2], double complex** *b***[2][2],**

const double complex *z0*[2]);

void vnaconv_utos(const double complex *u*[2][2], **double complex** *s*[2][2]);

void vnaconv_utot(const double complex *u*[2][2], **double complex** *t*[2][2]);

void vnaconv_utoz(const double complex *u*[2][2], **double complex** *z*[2][2],
  const double complex *z0*[2]);

void vnaconv_utoy(const double complex *u*[2][2], **double complex** *y*[2][2],
  const double complex *z0*[2]);

void vnaconv_utoh(const double complex *u*[2][2], **double complex** *h*[2][2],
  const double complex *z0*[2]);

void vnaconv_utog(const double complex *u*[2][2], **double complex** *g*[2][2],
  const double complex *z0*[2]);

void vnaconv_utoa(const double complex *u*[2][2], **double complex** *a*[2][2],
  const double complex *z0*[2]);

void vnaconv_utob(const double complex *u*[2][2], **double complex** *b*[2][2],
  const double complex *z0*[2]);

void vnaconv_ztos(const double complex *z*[2][2], **double complex** *s*[2][2],
  const double complex *z0*[2]);

void vnaconv_ztot(const double complex *z*[2][2], **double complex** *t*[2][2],
  const double complex *z0*[2]);

void vnaconv_ztou(const double complex *z*[2][2], **double complex** *u*[2][2],
  const double complex *z0*[2]);

void vnaconv_ztoh(const double complex *z*[2][2], **double complex** *h*[2][2]);

void vnaconv_ztog(const double complex *z*[2][2], **double complex** *g*[2][2]);

void vnaconv_ztoa(const double complex *z*[2][2], **double complex** *a*[2][2]);

void vnaconv_ztob(const double complex *z*[2][2], **double complex** *b*[2][2]);

void vnaconv_ytos(const double complex *y*[2][2], **double complex** *s*[2][2],
  const double complex *z0*[2]);

void vnaconv_ytot(const double complex *y*[2][2], **double complex** *t*[2][2],
  const double complex *z0*[2]);

void vnaconv_ytou(const double complex *y*[2][2], **double complex** *u*[2][2],
  const double complex *z0*[2]);

void vnaconv_ytoh(const double complex *y*[2][2], **double complex** *h*[2][2]);

void vnaconv_ytog(const double complex *y*[2][2], **double complex** *g*[2][2]);

void vnaconv_ytoa(const double complex *y*[2][2], **double complex** *a*[2][2]);

void vnaconv_ytob(const double complex *y*[2][2], **double complex** *b*[2][2]);

void vnaconv_htos(const double complex *h*[2][2], **double complex** *s*[2][2],
  const double complex *z0*[2]);

void vnaconv_htot(const double complex *h*[2][2], **double complex** *t*[2][2],
  const double complex *z0*[2]);

void vnaconv_htou(const double complex *h*[2][2], **double complex** *u*[2][2],
  const double complex *z0*[2]);

void vnaconv_htoz(const double complex *h*[2][2], **double complex** *z*[2][2]);

void vnaconv_htoy(const double complex *h*[2][2], **double complex** *y*[2][2]);

void vnaconv_htog(const double complex *h*[2][2], **double complex** *g*[2][2]);

void vnaconv_htoa(const double complex *h*[2][2], **double complex** *a*[2][2]);

void vnaconv_htob(const double complex *h*[2][2], **double complex** *b*[2][2]);

void vnaconv_gtos(const double complex *g*[2][2], **double complex** *s*[2][2],
    const double complex *z0*[2]);

void vnaconv_gtot(const double complex *g*[2][2], **double complex** *t*[2][2],
    const double complex *z0*[2]);

void vnaconv_gtou(const double complex *g*[2][2], **double complex** *u*[2][2],
    const double complex *z0*[2]);

void vnaconv_gtoz(const double complex *g*[2][2], **double complex** *z*[2][2]);

void vnaconv_gtoy(const double complex *g*[2][2], **double complex** *y*[2][2]);

void vnaconv_gtoh(const double complex *g*[2][2], **double complex** *h*[2][2]);

void vnaconv_gtoa(const double complex *g*[2][2], **double complex** *a*[2][2]);

void vnaconv_gtob(const double complex *g*[2][2], **double complex** *b*[2][2]);

void vnaconv_atos(const double complex *a*[2][2], **double complex** *s*[2][2],
    const double complex *z0*[2]);

void vnaconv_atot(const double complex *a*[2][2], **double complex** *t*[2][2],
    const double complex *z0*[2]);

void vnaconv_atou(const double complex *a*[2][2], **double complex** *u*[2][2],
    const double complex *z0*[2]);

void vnaconv_atoz(const double complex *a*[2][2], **double complex** *z*[2][2]);

void vnaconv_atoy(const double complex *a*[2][2], **double complex** *y*[2][2]);

void vnaconv_atoh(const double complex *a*[2][2], **double complex** *h*[2][2]);

void vnaconv_atog(const double complex *a*[2][2], **double complex** *g*[2][2]);

void vnaconv_atob(const double complex *a*[2][2], **double complex** *b*[2][2]);

void vnaconv_btos(const double complex *b*[2][2], **double complex** *s*[2][2],
    const double complex *z0*[2]);

void vnaconv_btot(const double complex *b*[2][2], **double complex** *t*[2][2],
    const double complex *z0*[2]);

void vnaconv_btou(const double complex *b*[2][2], **double complex** *u*[2][2],
    const double complex *z0*[2]);

void vnaconv_btoz(const double complex *b*[2][2], **double complex** *z*[2][2]);

void vnaconv_btoy(const double complex *b*[2][2], **double complex** *y*[2][2]);

void vnaconv_btoh(const double complex *b*[2][2], **double complex** *h*[2][2]);

void vnaconv_btog(const double complex *b*[2][2], **double complex** *g*[2][2]);

void vnaconv_btoa(const double complex *b*[2][2], **double complex** *a*[2][2]);

2-Port Matrix to Input Impedance

void vnaconv_stozi(const double complex *s*[2][2], **double complex** *zi*[2],
    const double complex *z0*[2]);

void vnaconv_ttozi(const double complex *t*[2][2], **double complex** *zi*[2],
    const double complex *z0*[2]);

void vnaconv_utozi(const double complex *u*[2][2], **double complex** *zi*[2],
    const double complex *z0*[2]);

void vnaconv_ztozi(const double complex *z***[2][2], double complex** *zi***[2],**
const double complex *z0***[2]);**

void vnaconv_ytozi(const double complex *y***[2][2], double complex** *zi***[2],**
const double complex *z0***[2]);**

void vnaconv_htozi(const double complex *h***[2][2], double complex** *zi***[2],**
const double complex *z0***[2]);**

void vnaconv_gtozi(const double complex *g***[2][2], double complex** *zi***[2],**
const double complex *z0***[2]);**

void vnaconv_atozi(const double complex *a***[2][2], double complex** *zi***[2],**
const double complex *z0***[2]);**

void vnaconv_btozi(const double complex *b***[2][2], double complex** *zi***[2],**
const double complex *z0***[2]);**

N-Port Matrix Conversions

void vnaconv_stozn(const double complex ****s*, double complex ****z***,**
const double complex ****z0*, int** *n***);**

void vnaconv_stoyn(const double complex ****s*, double complex ****y***,**
const double complex ****z0*, int** *n***);**

void vnaconv_ztosn(const double complex ****z*, double complex ****s***,**
const double complex ****z0*, int** *n***);**

void vnaconv_ztoyn(const double complex ****z*, double complex ****y***, int** *n***);**

void vnaconv_ytosn(const double complex ****y*, double complex ****s***,**
const double complex ****z0*, int** *n***);**

void vnaconv_ytozn(const double complex ****y*, double complex ****z***, int** *n***);**

N-Port Matrix To Input Impedance

void vnaconv_stozin(const double complex ****s*, double complex ****zi***,**
const double complex ****z0*, int** *n***);**

void vnaconv_ztozin(const double complex ****z*, double complex ****zi***,**
const double complex ****z0*, int** *n***);**

void vnaconv_ytozin(const double complex ****y*, double complex ****zi***,**
const double complex ****z0*, int** *n***);**

DESCRIPTION

These functions convert between various mathematical representations of electrical n-port networks. Representations include scattering (s-parameters), scattering-transfer (t-parameters), inverse scattering-transfer (u-parameters), impedance (z-parameters), admittance (y-parameters), hybrid (h-parameters), inverse hybrid (g-parameters), ABCD (a-parameters) and inverse ABCD (b-parameters).

While s-parameters, z-parameters and y-parameters are defined for any number of ports, t-parameters, u-paramters, h-parameters, g-parameters, a-parameters and b-parameters are defined for two-port networks only. The library contains one set of functions for two-port networks and another set of functions for n-port networks – the later all have names ending in **n**. For example, **vnaconv_stoy**() is the function to convert from s-parameters to y-parameters for two-port, while **vnaconv_stoyn**() is the equivalent function for n-ports. The two-port functions take matrices of type double complex [2][2] while the n-port functions take the address of the first element of an *n* x *n* complex matrix (appearing in memory in C row-major order). In both cases, the input and output matrices can refer to the same memory, i.e. you can pass the same matrix as input and output to do an in-place conversion.

The *z0* parameter, common to both cases, is a pointer to a vector of system impedances, i.e. the impedance seen by the network looking out of each of its ports.

Two-port example:

```
    double complex s[2][2];
    double complex z[2][2];
    static double complex z0[2] = { 50.0, 50.0 };

    vnaconv_stoz(s, z, z0);
```

N-port example:

```
    double complex s[3][3];
    double complex z[3][3];
    static double complex z0[3] = { 50.0, 75.0, 110.0 };

    vnaconv_stozn(&s[0][0], &z[0][0], z0, 3);
```

In order to give a more detailed description of the various parameter matrices, we must first give a few definitions:

a1 and a2 are the incident voltages into ports 1 and 2,
b1 and b2 are the reflected voltages out of ports 1 and 2,
v1 and v2 are the voltages at ports 1 and 2,
i1 and i2 are the currents into ports 1 and 2, and
Z1 and Z2 are the system impedances the device sees looking out of its ports.

Note that for a1, a2, b1 and b2, we're using "voltage" loosely. More precisely, these values are defined as root power in units of Watt^(1/2). In most cases, this distinction isn't important because the scale factor divides out.

The relationships between A_i, B_i, V_i, I_i are:

$$a_i = \frac{1}{2} K_i(V_i + I_i Z_i) \qquad V_i = \frac{a_i Z_i^* + b_i Z_i}{K_i re(Z_i)}$$

$$b_i = \frac{1}{2} K_i(V_i - I_i Z_i^*) \qquad I_i = \frac{a_i - b_i}{K_i re(Z_i)}$$

where $K_i = \dfrac{1}{\sqrt{|re(Z_i)|}}$, and * is the conjugation operator.

We can now show the relationships for each representation of network parameters. The **s** (scattering) parameters satisfy:

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

The **t** (scattering-transfer) parameters satisfy:

$$\begin{bmatrix} b_1 \\ a_1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$

The **t** parameters for a cascade of two-port networks is the left-to-right matrix product of the **t** parameters of each successive stage.

The **u** (inverse scattering-transfer) parameters satisfy:

$$\begin{bmatrix} a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} b_1 \\ a_1 \end{bmatrix}$$

The **u** parameters for a cascade of two-port networks is the right-to-left matrix product of the **u** parameters of each successive stage.

The **z** (impedance) parameters satisfy:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

The **y** (admittance) parameters satisfy:

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

The **h** (hybrid) parameters satisfy:

$$\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix}$$

The **g** (inverse hybrid) parameters satisfy:

$$\begin{bmatrix} I_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ I_2 \end{bmatrix}$$

The **a** (ABCD) parameters satisfy:

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix}$$

The **a** parameters for a cascade of two-port networks is the left-to-right matrix product of the **a** parameters of each successive stage. Don't confuse the **a** matrix with the a1 and a2 voltages above.

The **b** (inverse ABCD) parameters satisfy:

$$\begin{bmatrix} V_2 \\ -I_2 \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ I_1 \end{bmatrix}$$

The **b** parameters for a cascade of two-port networks is the right-to-left matrix product of the **b** parameters of each successive stage. Don't confuse the **b** matrix with the b1 and b2 voltages above.

In addition to the functions that convert between parameter forms, there are also functions that calculate the input impedances looking into each port of the network when all other ports are terminated at the system impedances. For example, **vnaconv_stozi**() finds the input impedances from a 2x2 s-parameter matrix, while **vnaconv_ztozin**() finds the input impedances from an *n* by *n* z-parameter matrix. The *zi* and *z0* vectors must have length *n*.

RETURN VALUE

All functions return void. The result matrix may contain inf or nan values if the conversion is nondeterministic.

NOTES

The declarations showing parameters with array types, e.g. **double complex** *s*[2][2], would be more honestly written as having type pointer to an element of the array. In this particular example the actual type is **double complex** (**s*)[2] – pointer to array of two double complex values. If you have a variable of this type and you add 1 to it, it advances to the next row. We've chosen, however, to use the equivalent in C (but somewhat misleading) parameter array notation to more clearly document what the user is expected to pass to the function. In the include file, they're written in the pointer to element form.

EXAMPLES

```
#include <complex.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <vnaconv.h>

/* system impedances */
#define Z1      75.0
#define Z2      50.0

/* resistor values for impedance matching L pad */
#define R1      (sqrt(Z1) * sqrt(Z1 - Z2))
#define R2      (sqrt(Z1) * Z2 / sqrt(Z1 - Z2))

/* system impedance vector */
static const double complex z0[] = { Z1, Z2 };

int main(int argc, char **argv)
{
    const double complex z[2][2] = { /* Z-parameters of the L pad */
        { R1+R2, R2 },
        { R2,    R2 }
    };
    double complex s[2][2];
    double complex zi[2];

    /*
     * Convert to S-parameters.
     */
    vnaconv_ztos(z, s, z0);
    (void)printf("s-parameters:\n");
    (void)printf("  %7.4f%+7.4fi    %7.4f%+7.4fi\n",
        creal(s[0][0]), cimag(s[0][0]), creal(s[0][1]), cimag(s[0][1]));
    (void)printf("  %7.4f%+7.4fi    %7.4f%+7.4fi\n",
        creal(s[1][0]), cimag(s[1][0]), creal(s[1][1]), cimag(s[1][1]));
    (void)printf("\n");

    /*
     * Convert to input impedance at each port.
     */
    vnaconv_stozi(s, zi, z0);
    (void)printf("input-impedances:\n");
    (void)printf("  %7.4f%+7.4fi    %7.4f%+7.4fi\n",
        creal(zi[0]), cimag(zi[0]), creal(zi[1]), cimag(zi[1]));
    (void)printf("\n");
```

```
            exit(0);
        }
```

SEE ALSO

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