# TABLE OF CONTENTS
# SECTION 7.0

# SECTION 7.0
# MODAL ANALYSIS FILE CONVERSION

## Introduction

One common application of Dynamic Signal Analyzers is the collection of data for use in Modal Analysis. Modal Analysis is generally defined as follows:

*The determination of an optimal set of parameters that can be used to describe the dynamic response of a mechanical system to various initial conditions and/or forced excitations*[1].

These "Modal Parameters" are often represented graphically as the "Mode Shapes" of the system: the shape assumed by the system at maximum deflection when excited at a resonance (natural frequency). Mathematically the resonance will be the eigenvalues of the system and the mode shapes will be the eigenvectors.

One of the properties of a linear time invariant mechanical system is that any vibration of the system can be described by a linear combination of the of the mode shapes. Therefore, given the modal parameters, we can predict the response of the system to any excitation.

## Modal Test Example

Our test body would be a mechanical structure of some kind, either freely suspended or rigidly supported. Upon this structure would be one or more strategically mounted accelerometers. The structure would then typically be excited using either a sinusoidal or impulsive load. Sinusoidal input would typically take the form of connecting the structure to a mechanical shaker and sweeping it up and/or down the frequency range of interest. Impulsive loading would generally involve striking the structure with a special "Impulse" hammer that is equipped with a load cell to determine the energy entered into the system. In either case, both the excitation and response are measured, and from these a "Frequency Response Function" (modal terminology for a Transfer Function), or FRF, is calculated. From (1) the geometry of the structure, (2) the FRFs and (3) the geometric relationship between the two determined by the placement of the accelerometers, it is possible to empirically determine the "modes of vibration."

---

[1] Dave Brown, *Lecture Notes from "Modal Analysis: theory and Application Course,"* IMAC XVI, Santa Barbara, CA, Jan. 29-31, 1998,
p. 3

# Using SigLab in Modal Analysis

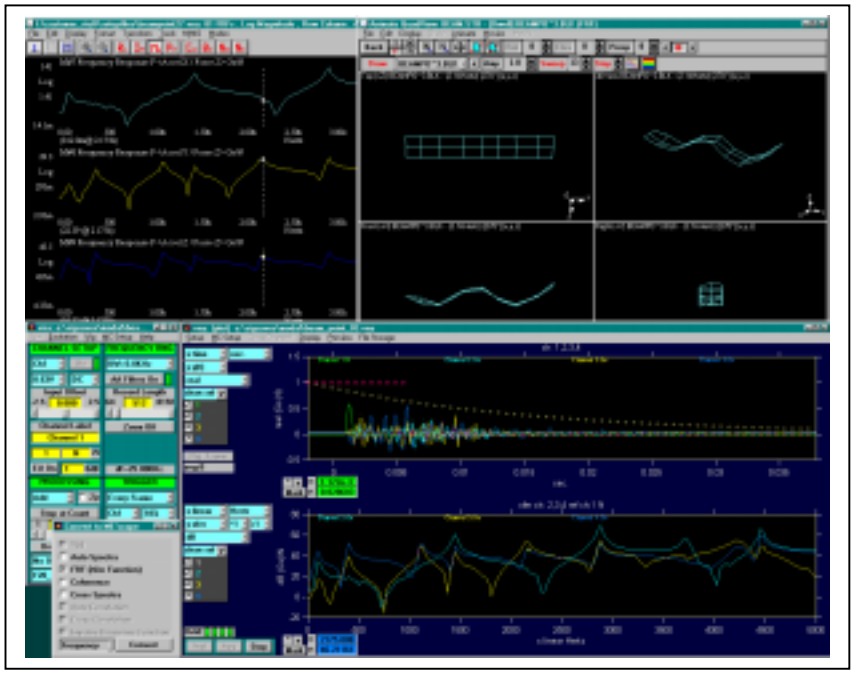

***Figure 7-1 An example of a SigLab modal test. The structure used was a free-free aluminum beam approximately 7" in length, 1.5" wide and 1/8" thick. A triaxial accelerometer was mounted at 1 end. Using VNA, with a modal impact hammer to excite the beam, 3 FRF measurements were made at each of 27 points on the beam. These were then imported into ME'scope and used to animate a wireframe model of the beam.***

SigLab is well suited to provide the FRFs needed as input into Modal Analysis. It offers exceptional measurement quality on up to 16 channels, force and exponential windows and integrated bias source for powering accelerometers and impulse hammers without the need of an external amplifier. Depending on the measurement technique desired, either ***vna*** or ***vss*** can generate quality FRF measurements.

You then must do Modal Analysis on these FRF measurements. Although MATLAB has the power you need to "roll your own" modal analysis package, given the number of commercial packages available, there is little need for you to do so.

However, since most of these packages are not implemented in MATLAB[2], your software package includes software routines to convert the ***vna*** or ***vss*** data stored in MATLAB format into a form readable by most commercial Modal Analysis packages.

SigLab files can be translated into any of three formats. Two of the most popular and powerful Modal Analysis packages on the market today are ME'scope from Vibrant Technology Inc. and STARModal from Spectral Dynamics. Given the prevalence of these

[2] Exceptions to this rule would be the "Structural Dynamics Toolbox," currently available from Scientific Solutions and the "X-modal" package being developed at the University of Cincinnati Structural Dynamics Lab.

two packages, routines are provided to translate SigLab files directly into their respective file formats. For users of other packages, routines are provided to translate SigLab files into Universal File Format (UFF) ASCII files, a standard file format (as the name suggests) which can be read by any Modal Analysis package.

# SigLab to ME'scope

If you are using the ME'scope modal analysis package from Vibrant Technology Inc., you have several different options for getting your data from MATLAB into ME'scope. ME'scope files can be written directly out of ***vna*** as a File Menu selection or called from the command prompt. If you have written your own code you can write your own ME'scope translation utilities directly using a MATLAB MEX function.

## *Writing ME'scope Files Directly from VNA*

This is a fairly straightforward and easy procedure. Simply select from the ***vna*** *File* Menu the option "*Save As to ME'scope File*." This will call up the following figure:

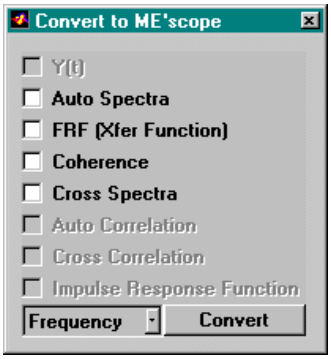


***Figure 7-2 Measurement Type to Convert to ME'scope Selection Panel. Available measurement types (based on the Measurement Domain Selection popup menu at bottom left) are shown in bold. All other measurement types are greyed out.***

From here, you can select from 8 measurement types, divided between 4 classes of measurements. You can only select measurements from a single class. The class you select is determined from the Measurement Domain Selection popup menu at the lower left of the figure. The four classes are:

1. Frequency Domain: Auto Spectra, FRF, Coherence and Cross Spectra measurements.

2. Time Domain: y(t) measurements.

3. Impulse Response Domain: Impulse Response Functions.

4. Correlation Domain: Auto Correlation and Cross Correlation measurements.

The reason for the limitation comes from the type of file you are writing. The ME'scope Data Block file (*.BLK files can contain as many traces as you desire (the limitation of 2048 traces is impossible to reach from a single ***vna*** file) with a maximum length of 1 million samples per trace (again, impossible within ***vna***). The only limitation is that all traces in a given block file must use the exact same X-axis vector.

Once you have selected the measurement types you wish to convert, click on the *Convert* pushbutton at the lower right. A dialog box will open prompting you to define a new BLK file to write. Then a message box appears advising you that the measurements were successfully sent to the DLL and that the BLK file was successfully written. If for some reason you were unsuccessful (for example, you can not overwrite an existing BLK file), these message boxes will inform you and supply the reason for the failure.

## *Writing VNA and VSS Files to ME'scope*

The MATLAB function, **vna2mesc,** called by the "*Save As a ME'scope File*" menu selection in ***vna*** can also be called directly at the MATLAB prompt. This can take from 0 to 3 input arguments and has the following usages:

1. When you type at the matlab command prompt:

```
>> vna2mesc
```

   if ***vna*** is open, the data will be pulled out of ***vna***;
   if not, you will be prompted to select a ***vna*** file.
   In either case the user will be prompted for the BLK file to be written.

2. To specifically pull the data out of ***vna*** (must be open), use either

```
>> vna2mesc('active');
```

   in which case you will be prompted for the BLK file to be written , or specify the BLK file in a string, BlockFile:

```
>> vna2mesc('active',BlockFile);
```

3. To specifically pull the data out of a VNA file, use

```
>> vna2mesc('file');
```

   in which case you will be queried for both the BLK and ***vna*** files. Otherwise you can specify just the BLK file with

```
>> vna2mesc('file',BlockFile);
```

   and get prompted for the VNA file, or else specify both files using:

```
>> vna2mesc('file',BlockFile,VnaFile);
```

In form where the input argument ***VnaFile*** is specified, ***VnaFile*** can take three forms:

1. a single string containing a ***vna*** filename

2. a Matrix of strings, where each row is a ***vna*** filename

3. a cell array, where every cell is a string containing a ***vna*** filename.

In the case of multiple filenames (2 and 3 above), all ***vna*** files must have the same X-axis values. If later X-axes do NOT match the first file, then that file will be skipped.

The following code is an example of a short m-file used to create a 27-element cell array from the files beampoint01.vna through beampoint27.vna (see Figure 7-1):

```
for k = 1:27
  if k<10
    VnaFile{k} = ['f:\setupfiles\beampoint0',...
                  int2str(k),'.vna'];
  else
    VnaFile{k} = ['f:\setupfiles\beampoint',...
                  int2str(k),'.vna'];
  end;
end;
BlkFile = 'f:\ setupfiles\bpoint3.blk';
vna2mesc('file',BlkFile,VnaFile);
```

For those unfamiliar with the use of cell arrays in MATLAB, it is important to note the use of curly brackets, { }, rather than normal parentheses, ( ).

**Vna2mesc.m** will not work on older ***vna*** files that predate version 3.0 of the SigLab software. This is because the measurement data is stored differently (see the section on the SLm data structure in the Programming Guide for more information on how the new software stores the data). Therefore, three additional m-files are supplied which can be used from the command prompt to translate files stored in the older file format (<= v.2.24). These files are called **xferdat2mesc.m**, **aspecdat2mesc.m** and **timedat2mesc.m** respectively. These functions all take 2 input arguments. The first argument is a string containing the name of the SigLab file to be translated and the second a string containing the name of the ME'scope BLK file to write. If either argument is omitted, the user will be prompted with a dialog box. The first string will always be assumed to be the SigLab file and the second the ME'scope file, so you cannot omit the SigLab file but pass the ME'scope file. Unlike **vna2mesc.m**, these files do not support multiple SigLab files.

**Timedat2mesc.m** translates from ***vos, vsa*** or ***vna*** files stored in the old file format. It writes only the time data, y(t), to the BLK file. The default input file is .VOS, but you can type "*.vsa" or "*.vna" in the uigetfile "*File name*" to change the file type to that extension (the popup will only show the default type, m-files and mat-files).

**Aspecdat2mesc.m** will translate from ***vsa*** or ***vna*** files stored in the old file format. It writes only the autospectra measurements to the BLK file. The default file type if prompted is .VSA.

**Xferdat2mesc.m** is somewhat special. Not only does it translate from ***vna*** files stored in the old file format, it also supports any ***vss*** file (since the ***vss*** file structure has not changed with the release of SigLab v.3.0). The default file type if prompted is .VNA.

## *Writing your own MATLAB to ME'scope Translation Routines*

Vibrant Technologies distributes **Wrtblk32.dll,** a Windows DLL, to write ME'scope files. This DLL was copied into your \Windows\System during installation. DSPT provides a MATLAB MEX function, **sig2mesc32.dll,** that allows you to pass commands from MATLAB to **Wrtblk32.dll**. This MEX function is in your …\siglab\mod directory (you will also find a backup copy of **Wrtblk32.dll** in this directory).

If you wish to write your own MATLAB to ME'scope file utilities, you should refer to **WrtBlk32.doc** also in your …\siglab\mod. This document describes all the available functions in the **Wrtblk32.dll** library, including inputs, outputs and recommended calling order. In order to call these functions from within MATLAB, you would use the syntax:

```
sig2mesc32('WrtBlkFunctionName',Input1,Input2,…,InputN)
```

For example:

```
SetTraceSampleValue(TraceNum,…
      SampNum,RealVal,ImagVal)
```

becomes

```
sig2mesc32('SetTraceSampleValue',TraceNum, …
      SampNum,RealVal,ImagVal)
```

and

```
SetDataBlockXaxisStart(XaxisStart)
```

becomes

```
sig2mesc32('SetDataBlockXaxisStar',XaxisStart)
```

In addition to all the WrtBlk function calls, there are two other calls to **sig2mesc32.dll**:

1. **sig2mesc32('initdll')** initializes the DLL and establishes the links to the functions in Wrtblk32.dll. This function must be called before doing anything else.

2. **sig2mesc32('cleardll')** terminates the link with Wrtblk32.dll and should be called when you are done.

## SigLab to STARModal

Translating ***vna*** files to STARModal files is done using the UFF: Modal File Converter program, **uff.m**, and selecting the STAR entry in the File format popup menu:

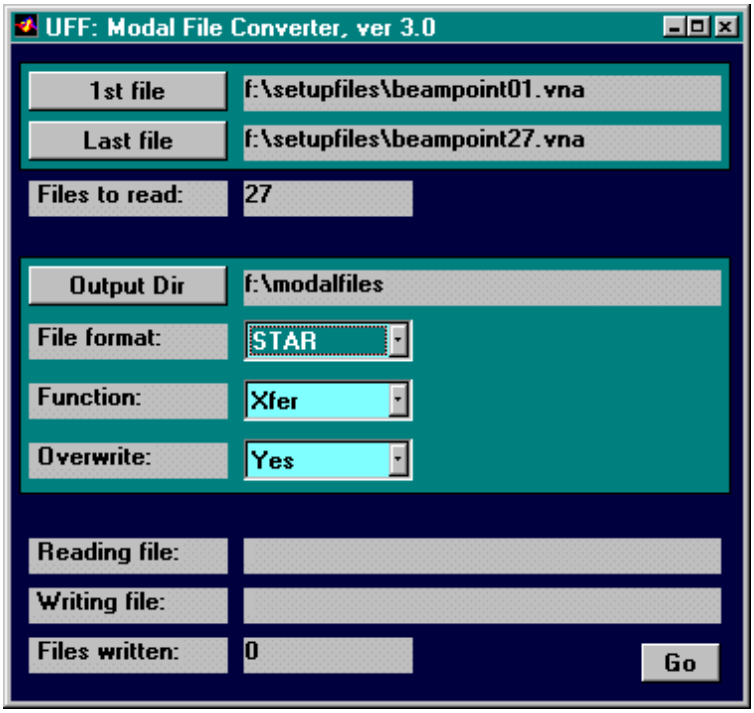


***Figure 7-3 UFF: Modal File Converter GUI with File format set to STARModal format.***

The files to be translated are determined by the *1st file* and *Last file* text fields. To set these files, Click on either the 1st file and Last file pushbuttons to call up a *uigetfile* dialog box and select the corresponding file. **Only *vna* files can be translated using this program**, and they must all share the same filename root, differentiated by having two or more digits as the end of the filename (see Figure 7-3). This takes advantage of the ***vna*** feature which auto-increments filenames that take this form. When the files are translated to STARModal format, therefore, 1st file, Last file and all the corresponding files in-between will be translated into STARModal files. The program can handle sequential gaps in the numbering of the filenames.

The following five measurement types can be translated into STARModal format:

1. FRFs (Xfer) [*.frf],
2. Crossspectra (Cspec) [*.cps],
3. Autospectra (Aspec) [*.aps],

4. Coherence (Coh) [*.coh], and

5. Time, y(t) (Time) [*.tim].

Only one measurement type may be selected per session of the program, although multiple sessions can be run sequentially to translate each measurement type. Support for multiple measurement types is superfluous because each STARModal file contains only a single trace. The measurement type determines (or, within STARModal, is determined by) the file extension (given in the list above within the square brackets [ ]). The Acronym contained in the parentheses in the list above is the selection from the Function Popup menu in UFF to select each measurement type.

STARModal has a very special naming convention for its filenames that determine with which point on the structure the measurement is associated. Because of this, it is necessary for ***vna*** and UFF to support this naming convention, which is accomplished through the ***vna*** Channel labels as follows:

When the structure is defined within STARModal, the points of the structure are identified by numbers. The measurements must be made along three principle axes:

1. x, y, and z in Cartesian coordinates

2. r, t and z in Cylindrical coordinates

3. r, p, and t in Spherical coordinates.

The points and directions (referred to as DOFs or degrees of freedoms in modal analysis lingo) at which the measurements are entered in the channel labels within ***vna*** are illustrated by the following example:

Suppose the excitation (measured by channel 1) is provided by a shaker attached to the structure at point 27 in the minus Z direction, and the response measured by channel 2 is provided by an accelerometer mounted at point 3 in the plus Y direction. Then, in ***vna,*** you would enter “-27z” or “-27Z” as the channel 1 label and “3y” or “3Y” as the channel 2 label. You may also put other information in the channel label as long as the DOF is first and you use a space as a separator. For example “-27Z shaker 2M” would be allowed. When the FRF from this file is translated into STARModal format, a file named 027z003y.frf is created. Note that channel 1 is always the reference and channels 2 and up are always response.

If the Channel Labels do not follow the above convention, a warning message will be displayed and a default ascending sequence of filenames will be used.

Channel labels that use this style are affected by the auto-increment routine on *File Save* (the same routine that increments filenames that end in two or more digits), as follows:

1. If a channel label begins with a 1 to 3 digit number (optionally with a leading + or -) followed by a direction code (one of the characters x, y, z, r, p or t) followed by a single comma, then the number will be incremented. For example. the channel label “78Y,” will be changed to “79Y,”.

2. If the direction code is followed by two consecutive commas then both the direction and point numbers will be incremented. For example the label “299x,,” on successive saves will be incremented to “299y,,”, “299z,,”, “300x,,”, “300y,,”, “300z,,”, etc.

3. If the channel label is not one of the above forms then that channel label will not be modified.

Click on the *Output dir* pushbutton to call up a dialog box to select where you would like to write the files. You will need to have at least one file in the target directory to select as a dummy file (but it will not be touched by the program).

The *Overwrite* popup menu gives you three choices to determine how the program should react if it encounters a file with the same name as the file it wishes to write. If you select “Yes”, files will be automatically overwritten. If you select “No”, the new file will not be written. If you select “Rename”, then the old file will be renamed to filename.old and then the new file written.

Once you have set up everything, click on the “*Go*” pushbutton. You will be kept appraised of your progress in the “*Reading file*:”, “*Writing file*” and “*Files written*” text fields.

## SigLab to Universal File Format

If you are using a Modal Analysis package other than ME'scope or STARModal, which is not MATLAB based, you will need to translate your data into the Universal File Format (UFF). SigLab supports the UFF-ascii format (there is also a UFF-binary format not supported by SigLab). UFF-ascii is supported by all Modal Analysis packages on the market today

In order to convert files to UFF-ascii, you use the same UFF: Modal File Converter program (**uff.m**) that you would use for STARModal, but set the File format to UFF-ascii. Otherwise, everything else works the same.

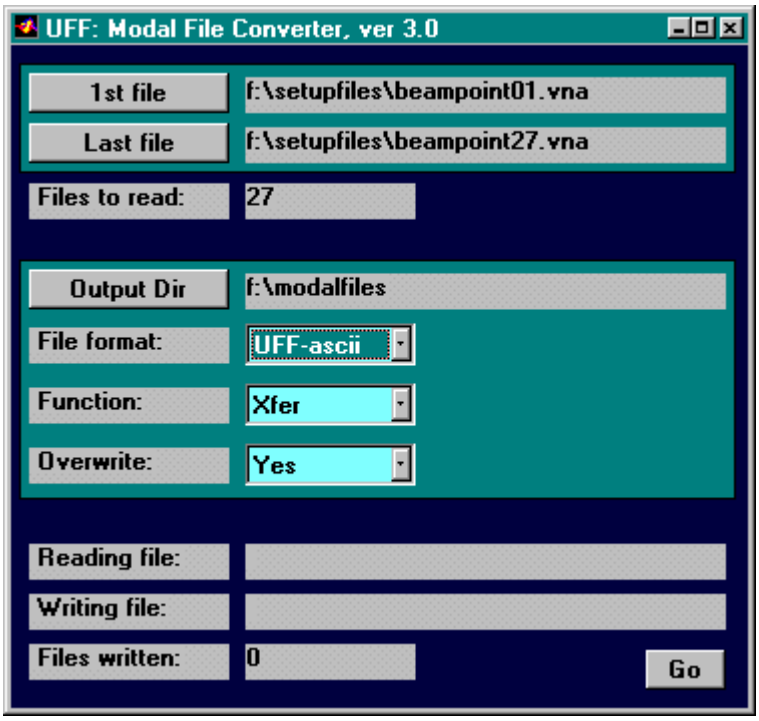


***Figure 7-4 UFF: Modal File Converter GUI with File format set to UFF-ascii format.***

The UFF files will be the extension "*.asc". Although the program will use the same naming convention for the root filename as was used for STARModal, this convention is not needed to read files by packages other than STARModal and the defaults are sufficient. However, since **uff.m** uses the same defaults every time it runs, you should probably consider adopting the STARModal naming system to keep track of your files.