TopSpin

ShapeTool

User Manual

Version 002

Innovation with Integrity

NMR

Document Number:

P/N: H146199

Contents

Contents

Contents

1 Introduction

1.1 About the ShapeTool

The Topspin ShapeTool allows you to create, analyze and manipulate RF- and Gradient Shapes in a fully parameterized manner.

The ShapeTool can be used in three different ways:

- The command **stdisp** opens a graphical interface.
- The command **st** gives access to the ShapeTool functionality from the TOPSPIN command line.
- The ShapeTool C interface allows the ShapeTool functionality to be called from inside AU programs.

The detailed workflow of the interactive ShapeTool is explained in *chapter 2 [▶ 11]*. *Chapters 3 [▶ 31]*, *4 [▶ 11]* and *5 [▶ 47]* contain information about the built-in functions for generating, manipulating and analyzing shapes. Examples are provided how to invoke these operations from the command line tool and through AU programs. *Chapters 6 [▶ 63]* and *7 [▶ 71]* contain information about further **st** commands and the use of the ShapeTool in AU programs including the documented C header file. *Chapter 8 [▶ 65]* gives a detailed look on the structure of shape files.

See also

Analyzing Shapes [▶ 55]

1.2 Shapes and Gradients

An RF shape consists of phase and amplitude values, whereas a gradient only consists of amplitude values.

A RF Shape/Gradient is a structure that consists of up to 16 arbitrary waveforms which are added together to give an overall waveform which is called a shape/gradient. For the remainder of this manual we will stick to the term shape if shape and gradient can be used interchangeably and if not pointed out otherwise.

For the generation of a waveform there exist several built in mathematical functions which can be controlled via a function specific set of parameters. Furthermore arbitrary user generated waveforms can be used. After generating a waveform it can be manipulated using several built in functions which again can be controlled by parameters. Shapes can be analyzed to provide characteristic values for further calculations, e.g. bandwith factor, integral factor etc.

Shapes and gradients can be converted into each other of course at the cost of losing the phase information when converting to a gradient.

The amplitude and phase values of the shape together with the parameters characterizing the shape are stored in text files in JCAMP-DX format suitable to be displayed with **stdisp** or executed by TopSpin acquisition commands.

Shape / Gradient files are stored in the directories

<topspinhome>/exp/stan/nmr/lists/wave

<topspinhome>/exp/stan/nmr/lists/wave/user

and

<topspinhome>/exp/stan/nmr/lists/gp

<topspinhome>/exp/stan/nmr/lists/gp/user

respectively with the wave and gp directory being read only which contain the TopSpin default shapes.

1.3 The interactive ShapeTool "stdisp"

The interactive Shapetool invoked by the command **stdisp** presents a graphical interface to view shapes, handle all generation, manipulation and analysis commands and to modify the shape parameters. Furthermore it allows to start NMR-SIM, calculate excitation profiles, setup sharc experiments and interacts with the ASED Parameters of the experiment. See *Chapter 2 [▷ 11]* for a detailed description.

1.4 The command line ShapeTool "st"

The following lists all Shapetool commands that can be invoked from the TOPSPIN command line. Expressions enclosed in <> have to be replaced by an actual value, expressions within [] are optional. Chapters 3-6 give detailed information about the possible operations.

Generate shape:

st generate <function> <size> [false] [filename=<outfile>] [-nobwcalc] <parameters>

Manipulate shape:

st manipulate <infile> [filename=<outfile>] <cmd> <parameters> [<freqlist>]

Analyze Shape:

st analyze <infile> <cmd> <parameters>

Add Shapes:

st add <alignment> <n> (<infile> <scale>)*n <outfile>

Merge Shapes:

st merge < infile1> <infile2> <outfile>

Convert from gradient to shape:

st convert <infile> <outfile> <shapeType> <#freq> [<deltaOmega*deltaT>]

Convert from shape to gradient:

st convertgrad <infile> <outfile>

1.5 The ShapeTool C interface for AU programs

To use the Shapetool functionality from inside AU programs you have to include the headerfile ShapeIOC.h into your AU program which after you can use the following routines:

Load shape into memory:

const double* readShapeC (

const char* fileName, int* nPoints, const double** phasePtr,

double* minAmp, double* maxAmp, double* integFac)

Write shape to disc:

int writeShapeC (
const char* fileName, const double* dataPtr, int nPoints,
int phaseData, double bwFac, int shapeMode)

Generate shape:

const double* generateShapeC (
const char* shapeName, const char* params, int phaseData)

Analyze shape:

int analyzeShapeC (
const char* fileName, const char* command, const char* params, char* result)

Manipulate shape:

const double* manipulateShapeC (
const char* fileName, const char* command, const char* params, const char* freqList, const double** phasePtr, int* nPoints, int* shapeMode)

Add shapes:

int addShapesC (
char* inputFile1, char* inputFile2, char* outputFile)

int addShapesMultC (
const char* inputFiles, const char* outputFile,
double* scaleF, int alignment)

Merge shapes:

int mergeShapesC (
char* inputFile1, char* inputFile2, char* outputFile)

int mergeShapesMultC (
const char* inputFiles, char* outputFile)

Get integral factor:

double getIntegFacC (
const char* fileName)

Delete shape from memory:

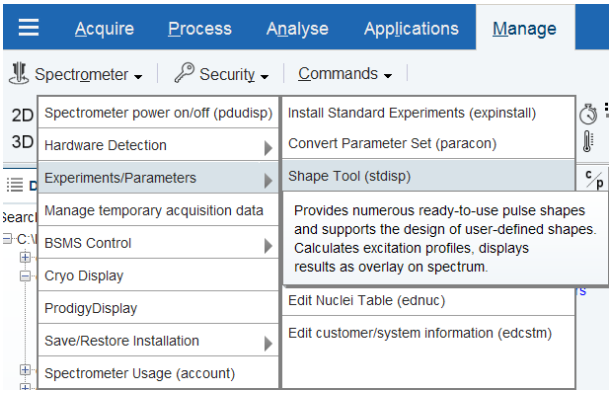
void deleteShapeC ()

Refer to Chapter 7 for the detailed function call declarations. Chapters 3-6 give detailed information about the possible operations.

2 Interactive ShapeTool

To start the interactive ShapeTool interface:

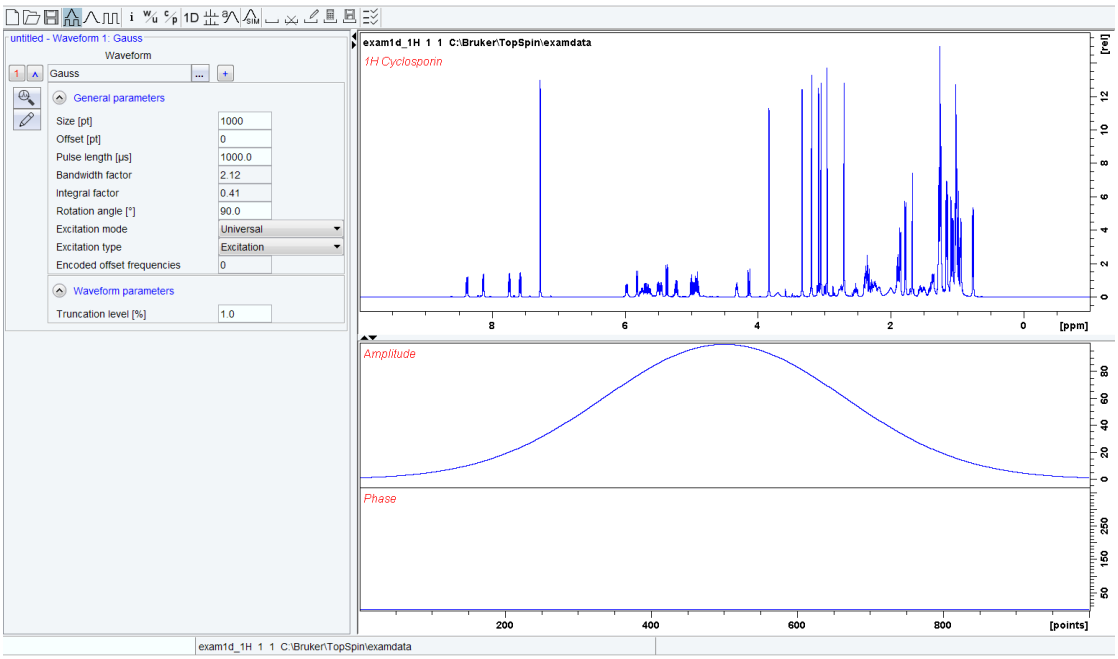
- Click **Manage | Spectrometer | Experiments/Parameters | ShapeTool (stdisp)**



or

- Enter **stdisp** on the command line.

The ShapeTool window will appear. This consists of a toolbar, a command line and a split pane with a data section at the right and a parameter section at the left.



By default, a 1000 point Gauss shape is displayed with Truncation level 1.0.

Note that all functions of the interactive ShapeTool can also be performed non-interactively with the TopSpin command **st**. This command must be entered with the appropriate arguments on the command line while the associated dataset is displayed and selected. For a description of the **st** command see:

? | Application manuals | Shape Tool | Chapter *3 to 6 [▶ 31]*

2.1 Opening a Shape/Gradient

To open an existing shape/gradient, click the button:

Open a shape or gradient.

Select **Shape** or **Gradient**, select a file from the appearing file list and click **OK**. The selected shape/gradient will appear in the data section.

To create a new shape click  (New Shape).

2.2 Using ShapeTool Display Options

The ShapeTool toolbar offers the following display options:

Display amplitude and phase.

Display amplitude only.

Display phase only.

Toggle cursor information on/off. Cursor information consists of the point number and amplitude/phase value.

Wrap/unwrap phase (actual phase/phase modulo 360°).

Toggle between polar and cartesian coordinates.

Dataset Interaction (for more details please refer to chapter *Setting Shape Tool Options [▷ 23]*)

2.3 Saving a Shape/Gradient as a 1D Dataset

To save the current shape/gradient stored under a procno of the associated dataset, click:

Save shape/gradient as a 1D dataset.

2.4 Superimposing Multiple Shapes/Gradients

Several shapes/gradient can be displayed superimposed using the following buttons:

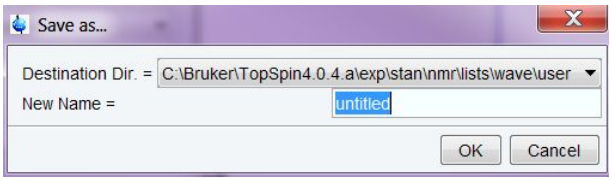
Add the currently displayed shape to multiple display.

Switch to multiple display mode. This shows all 'added' shapes/gradients superimposed.

2.5 Saving a Shape/Gradient

To save a shape/gradient, click the button:

Save a Shape or Gradient of it, and select **Shape, Gradient, Export to old format or as 1D**. A dialog window appears where you can enter a Destination Direction and New Name.



2.6 Generating Shapes/Gradients

The ShapeTool allows you to generate RF and gradients shapes. Various *Basic*, *Classical* or *Adiabatic* shapes can be created from the **Shapes** menu. Additionally *Solids*, *Imaging* and *Decoupling* shapes are available.

How to Generate a Gauss shape

To generate a Gauss shape:

Click ... | **Classical Shapes** | **Gauss**

A Gaussian shaped curve will appear in the data section. The parameter section shows the General parameters and Waveform parameters that can be set for a Gauss:

Pulse length, Rotation angle etc. It will also display the Bandwidth factor and the Integral factor.

Truncation level: The minimum amplitude at the edge of the shape.

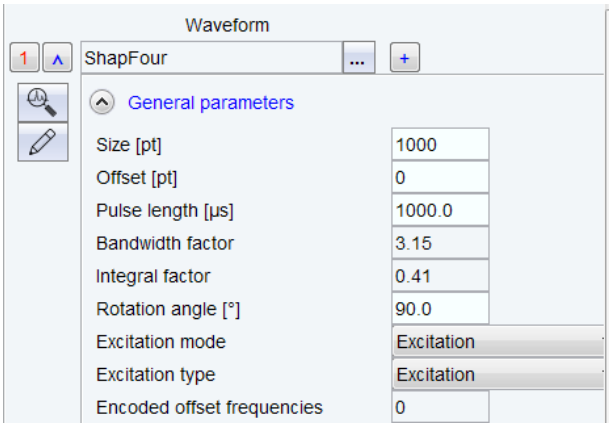
If you change these parameters, the displayed shape will automatically be updated. To save the shape, click , enter a *Name*, e.g. *mygauss* and *Title* (see *Saving a Shape/Gradient [▷ 12]*) and click **OK**.

How to Generate a ShapFour shape

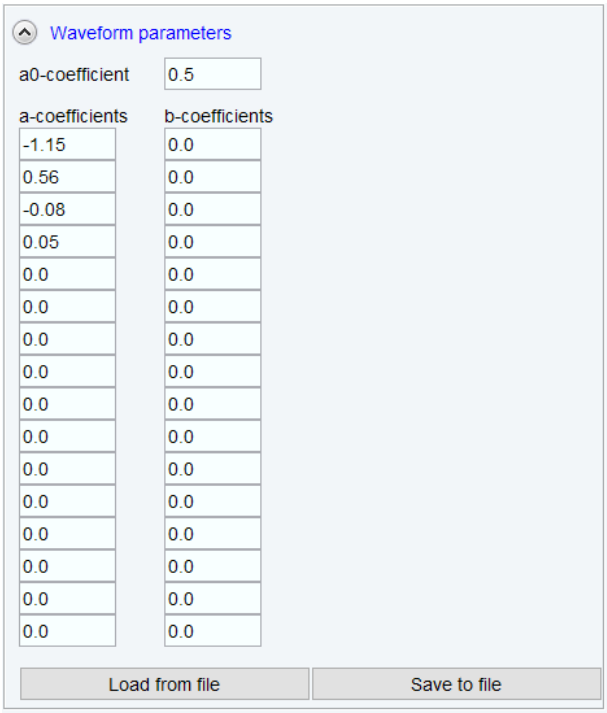
To generate an RF shape that is defined by Fourier coefficients:

Click ... | **Classical Shapes** | **ShapFour**

The following dialog box will appear:



Here you can enter the Size of Shape (General parameters) and the required coefficients (Waveform parameters). A ShapFour shape is defined by two coefficients arrays a[0,1,...] and b[1,...]. Note that the element a[0] is listed separately.



Here, you can enter the desired coefficients and then click:

Save to file to store the coefficients for later usage. You will be prompted for a filename which will be stored as:

<user-home>/.topspin-<hostname>/ShapeTool

With the coefficients above you have created a *Rsnob* shape.

To save it:

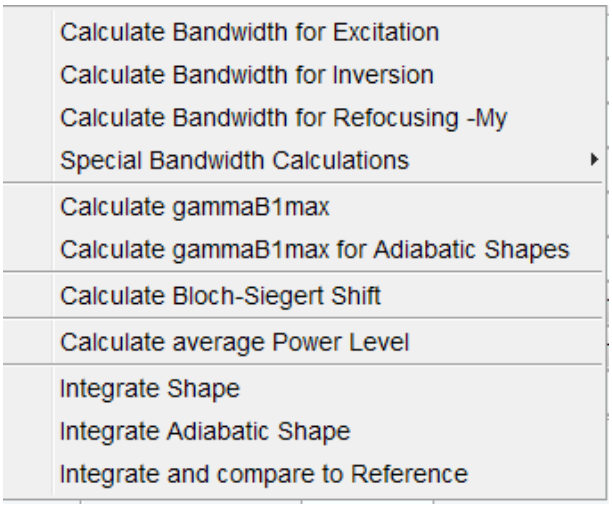
Click  , enter a *Name*, e.g. *myrsnob* and *Title* (see chapter *Saving a Shape/Gradient [▷ 12]*) and click **OK**.

2.7 Analyzing Shapes

The ShapeTool interface offers several functions to analyze shapes. Most of these functions are only meaningful for RF shapes.

To access these functions:

- Click  . This will open the pull-down menu shown:



From here, you can start the analysis functions discussed below.

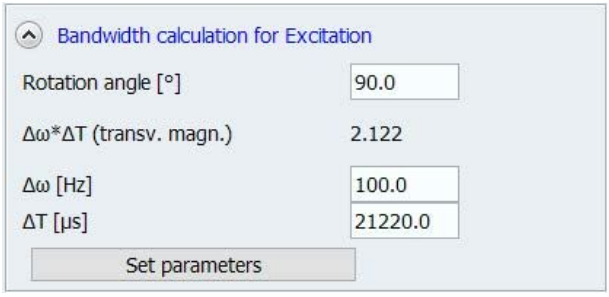
How to Calculate the Bandwidth for Excitation

To calculate the excitation band width factor Δω*ΔT:

Click  | **Calculate Bandwidth for Excitation**

For the excitation band width, the $\sqrt{(Mx^2 + My^2)}$ magnetization is used.

The following figure shows the resulting parameter section for a Gauss shape.



As you can see, for a 90° Gaussian shape the bandwidth factor is 2.122.

The bandwidth factor is the product of the width of excitation (DeltaOmega) and the pulse length (DeltaT).

This means a pulse length of 21220 µsec gives a band width (Δω) of 100.0 Hz. The bandwidth is the excitation width at the 3 dB point, i.e. the point where the magnetization has dropped to 70.8 %.

When you change one of the parameters in the parameter section, the others are automatically adjusted. Clicking  **(Dataset interaction) | Set parameters to ASED** asks to define the shape filename (eg. Gauss_100).

Clicking **OK** displays the pulse length and the power (default value, not yet calculated) and stores them in the parameters of the associated data set.

How to Calculate the Bandwidth for Inversion

To calculate the inversion band width factor Δω*ΔT :

Click  | **Calculate Bandwidth for Inversion**

For the inversion band width, the Mz magnetization is used.

How to Calculate Bandwidth for Refocusing

To calculate the refocusing band width factor Δω*ΔT:

Click  | **Calculate Bandwidth for Refocusing**

For the refocusing band width, the -My magnetization is used.

How to Calculate Special Bandwidths

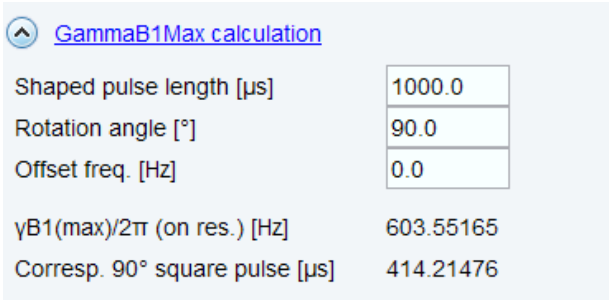
Further routines for the evaluation of various magnetization components are available under the menu item **Special Bandwidth Calculations**.

How to Calculate the Maximum RF Field Strength for Classical Pulses

To calculate the RF field strength for classical pulses:

Click  | **Calculate gammaB1max**

The following figure shows the resulting parameter section for a Gauss shape.



The result consists of the field strength (γB1max/2π) and the corresponding 90° square pulse length. To obtain this result, the integral of the shaped pulse is compared to that of a square pulse of same length, with the latter being normalized to 1.

The maximum power is:

γB1(max)/2π = γB1/2π (square pulse of same length) / integral ratio

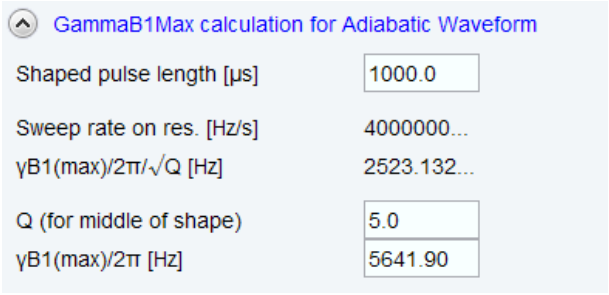
where

γB1/2π (square pulse of same length) = (1 / pulse length * (360°/ flip angle))

How to Calculate the Maximum RF Field Strength for Adiabatic Pulses

To calculate the RF field strength for adiabatic pulses:

Click  | **Calculate gammaB1max for Ad. shapes**



The figure shows the resulting parameter section for a *HypSec* shape.

The result γB1(max)/2π / √(Q) is calculated from the on-resonance sweep rate.

Q(t) is the quality or adiabaticity factor during the shape. After entering the appropriate value for Q with respect to the middle of the shape, the value for γB1(max)/2π is obtained. As a rule of thumb Q is set to 5 for inversion pulses. However, for decoupling pulses, Q should be set between 2 and 3 to allow a lower decoupling power.

ShapeTool always creates Adiabatic shapefiles corresponding to Q=5; in order to have a lower Q value the shapefile has to be created, then edited in Topspin and modified manually (parameter SHAPE_INTEGFAC).

How to Calculate the Bloch-Siegert Shift

To calculate the phase difference due to the Bloch-Siegert shift:

Click  | **Calculate Block-Siegert Shift**

Input parameters are the length of the shaped pulse in µs, γB1(max) in Hz and the offset of a theoretical signal relative to the frequency of the RF-pulse in Hz. The result consists of the phase difference due to Bloch-Siegert shift in Degree and the corresponding frequency shift in Hz.

How to Calculate the Average Power Level

To calculate the average power:

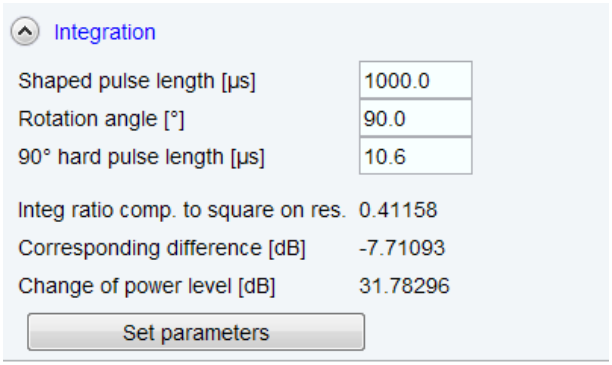
Click  | **Calculate average Power Level**

The result is the percentage of the average power of a square pulse of the same length.

How to Integrate Classical Shapes

To calculate the power level required for a classical shaped pulse:

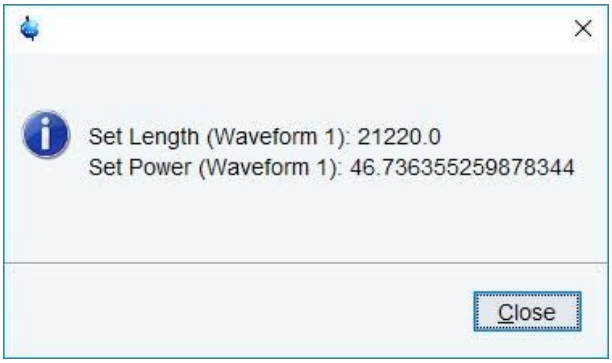
Click  => **Integrate Shape**



The figure above shows the parameter section where you can enter the length of the soft pulse, the flip angle and the length of the 90° hard pulse. The soft pulse length is initialized with the value updated at the end of the previous bandwidth calculation, otherwise a default value of 1000 µs is displayed. The hard pulse length is initialized with the value of the corresponding parameter (default: P1) of the associated dataset, if this value is 0 or if no dataset is associated a default value of 50 µs is displayed.

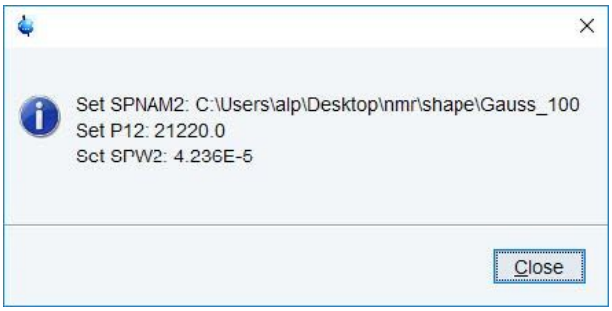
The result for a Gaussian shape with 1000 points and 1% truncation is shown. The *Integral ratio* and the *Corresponding difference* in dB are calculated from shape amplitude and phase, assuming equal pulse length and flip angle for soft and hard pulse. The *Change in power level,* however, is calculated from the specified pulse lengths and rotation.

Clicking on **Set parameters** (in the **Integrate Shape** window) calculates the power which is the sum of the shown **Change of power level [dB]** and the current hard pulse power level (in dBW, e.g. PLdB1). It is displayed (in dBW) together with the pulse length.



Clicking  | **Set Parameters to ased** asks to define the shape filename (ex. Gauss_100).

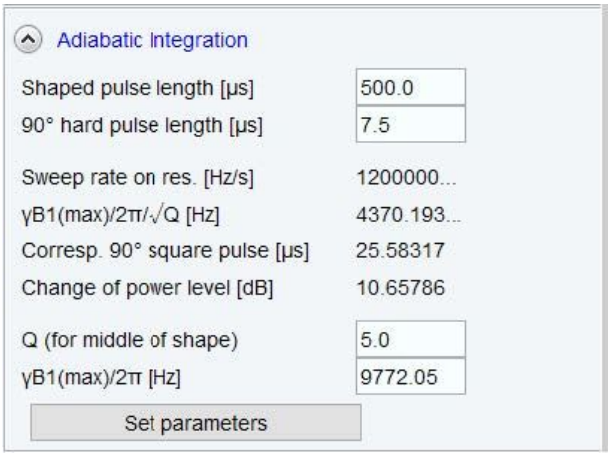
Clicking **OK** saves the length of the shaped pulse and the power level to the corresponding parameters of the associated dataset. Additionally, the values of the pulse length (µs) and the power (in W) are displayed.



How to Integrate Adiabatic Shapes

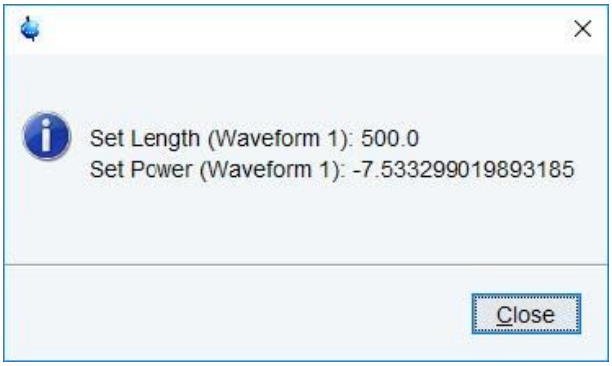
To calculate the power level required for an adiabatic shaped pulse:

Click  | **Integrate Adiabatic Shape**

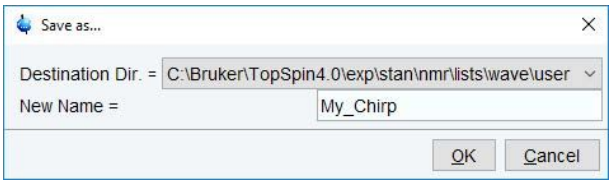


The change of power level is calculated from the length of the corresponding 90° the shape pulse length.

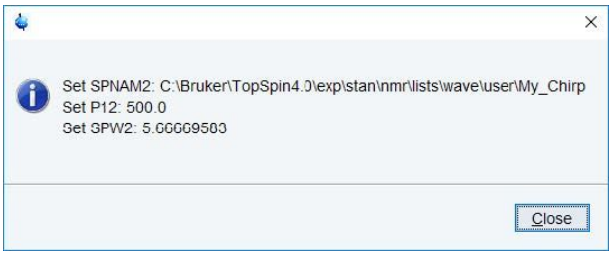
Clicking **Set parameters** (in the **Adiabatic Integration** window) calculates the power and displays it (in dBW) together with the pulse length.



Clicking  | **Set Parameters to ased** asks to define the shape filename (ex. My_Chirp).



Clicking **OK** saves the length of the shaped pulse and the change in required power level to the corresponding parameters of the associated dataset. Note that the latter is the sum of the shown Change of power level (in dB) and the current hard pulse power level (in dBW, e.g. PldB1). Additionally, the values of the pulse length (µs) and the power (in W) are displayed.



How to Start Simulation

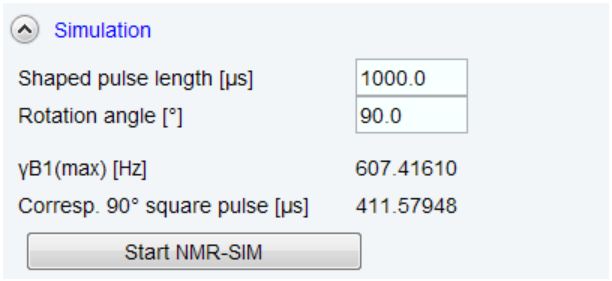
To evaluate the shape behavior, you can start the TopSpin NMRSIM routine:

Click 

.

This allows you for example, to view the excitation profile, or, for adiabatic pulses, check the adiabaticity.

An additional parameter field will show up where the required parameters of the shape can be entered (length, rotation angle):



Click Start NMR SIM to start simulation. The results will be shown in a separate window.

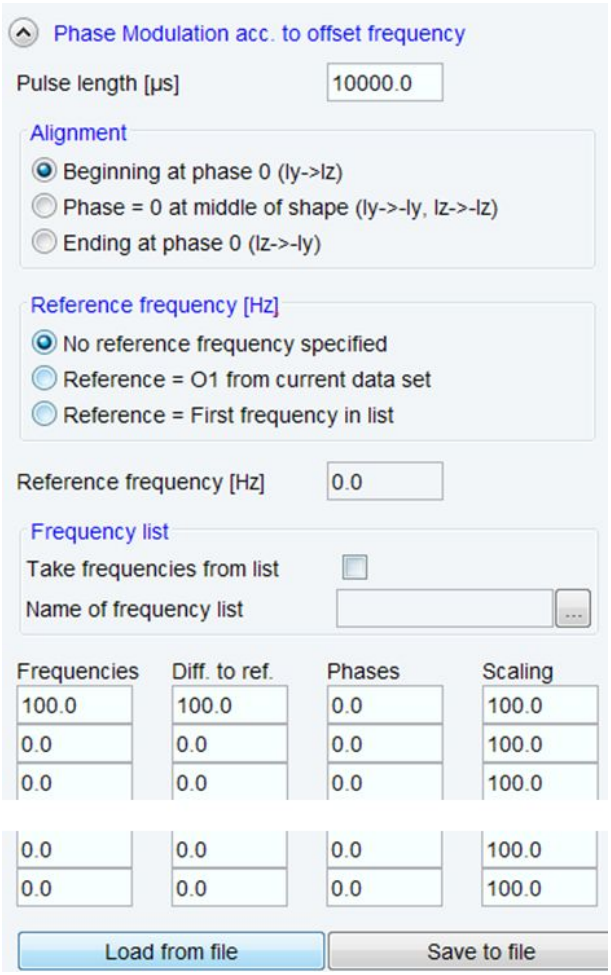
2.8 Manipulating Shapes

Shapes can be manipulated in various ways. Most of these functions are only meaningful for RF shapes.

Frequency Encoding

To perform a phase (and amplitude) modulation encoding one or more offset frequencies:

Click  | **Multiple Phase Modulation**



A dialog box will appear where you can set the required parameters.

The **Parameter field** shows the Pulse length [µs].

If no reference frequency is used, the values entered as frequencies have to be difference frequencies.

The **Alignment** radio buttons allow you to define the position of phase 0; at the beginning, in the middle or at the end of the shape (depends on the rotation type).

The **Reference Frequency** radio buttons allow you to choose whether the reference frequency:

- is not used
- is set to O1 of the associated dataset (will automatically update the Reference frequency field)
- is set to the first entry of the frequency list. The **Option** Take Frequencies from list will automatically be checked.

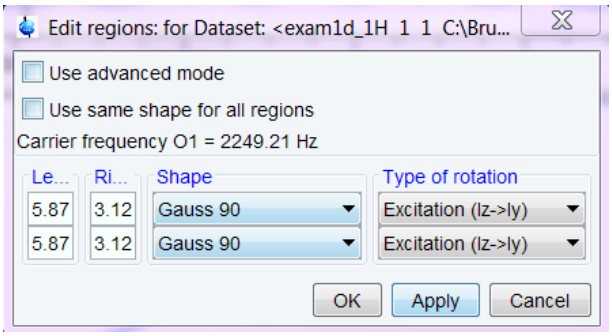
The **Frequency list** check boxes allow you to set:

- **Take Frequencies from list:** The Parameter Name of Frequency List allows to specify the name of the list. Frequency lists can be set up from the TopSpin interface with the command **edlist** or, interactively, by clicking the button.
- The table shows the entries for the frequencies, their differences to the reference frequency, an additional phase and a scaling factor (0-100%).

How to Calculate a Shape from an Excitation Region

A shape can be modulated according to specific regions in a spectrum. To do that, take the following steps:

- Click **New Excitation region** in the toolbar
- Determine the excitation regions interactively.
- To use the defined regions click .
- To edit all excitation regions click .



In the figure above the default values for Shape and Type of rotation will depend on the current General Parameters or associated waveforms in case of a combined shape. Depending on the application these settings have to be modified accordingly.

You can use the same shape for each region of all different shapes. In the latter case, the check box *Use same Shape for all Regions* must be unchecked.

How to Perform Single Sine Modulation

To perform an amplitude modulation such that the pulse excites at two symmetric sidebands (+/- offset) with opposite phase:

Click | **Sine Modulation** and set *Offset frequency* to be relative to the carrier frequency.

How to Perform Single Cosine Modulation

To perform an amplitude modulation such that the pulse excites at two symmetric sidebands (+/- offset) with the same phase:

Click | **Cosine Modulation** and set *Offset frequency* to be relative to the carrier frequency.

How to Create a Shape with a Linear Sweep

To create a shape with a phase modulation according to a linear frequency sweep:

Click | **Linear Sweep** and set the pulse length and sweep width. Then specify the Q factor to obtain the value for γB1(max)/2π. Q is the quality or adiabaticity factor which, as a rule of thumb Q is set to 5 for inversion pulses and between 2 and 3 for decoupling pulses.

How to Create a Shape with Const. Adiabaticity Sweep

To create a shape with a phase modulation according to a constant adiabaticity sweep:

Click  | **Const. Adiabaticity Sweep** and set the pulse length and sweep width. Then specify the Q factor to obtain the value for γB1(max)/2π. Q is the quality or adiabaticity factor which, as a rule of thumb Q is set to 5 for inversion pulses and between 2 and 3 for decoupling pulses.

How to create a Shape with Single Phase Modulation

Click  | Single Phase Modulation and set Pulse length, Offset, Offset phase alignment and Initial offset phase.

How to Calculate a Shape with Amplitude to the Power x

To create a shape with an amplitude to the power of a specified exponent:

Click  | **Power of Amplitude** and enter the desired exponent.

How to Add Constant Phase

To add a constant phase to the shape:

Click  | **Add constant Phase** and enter the phase to be added.

How to Perform Time Reversal

To time reverse a shape:

Click  | **Time Reversal**

How to Expand Shape

To expand a shape according to a specified phase list (supercycle):

Click  | **Expand Shape** and selected a phase cycle. The following standard phase cycles are offered:

mlev4, *mlev16*, *p5(150 deg)*, *p5(330 deg)*, *p5m4*, *p5m16*, *p9*, *p9m16*, *p5p9*

For example, **mlev4** has the following supercycle: 0 0 180 180

In addition, the entry YourOwn appears, which allows you to define your own phase cycles. These can be saved for later usage. The specified file must have the extension *.expand*. It is stored in the directory:

<user-home>/.topspin-<hostname>/ShapeTool

How to Use Fraction

Click  | Fraction to set Start and End of fraction.

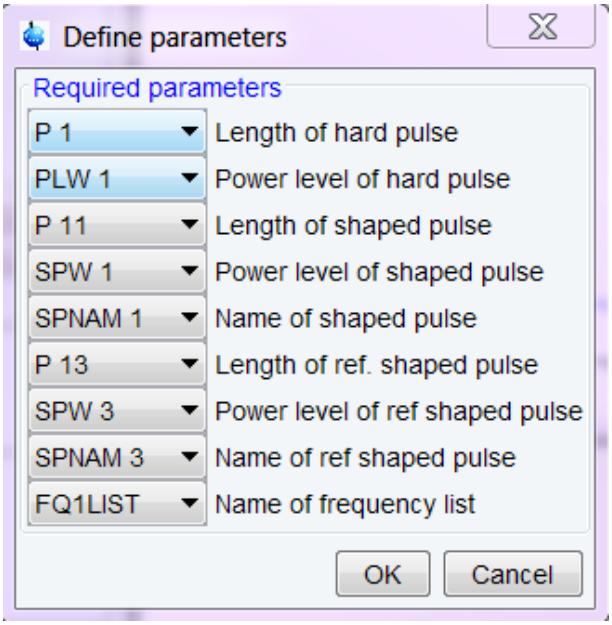
2.9 Setting ShapeTool Options

How to Change the Relation between ShapeTool and TopSpin parameters

To define the relation between ShapeTool parameters and TopSpin acquisition parameters:

- Click | **Define ASED parameters**

A dialog box will appear with a list box for each parameter.



Make any changes you want and click **OK**. The acquisition parameters of the associated dataset can be set/viewed from the TopSpin interface by clicking **AcquPars** or entering **eda**.

How to select the Associated Dataset

To set the dataset associated to the ShapeTool interface:

Click | **Select dataset**

A list of open datasets will appear. Select the desired dataset and click **OK**.

How to Get parameters from ASED

Click | **Get parameters from ASED.**

Length and Power of the associated dataset will be shown.

2.10 Examples of Using the ShapeTool

Example 1: Create a Shaped Pulse for Multiple Solvent Suppr. using WET

To create a shaped pulse:

- Acquire a reference 1D spectrum.
- Click to switch to frequency list mode.
  - Select the frequency list type and enter the list name *freqlist*. If you get a message that this list already exists, click **Overwrite** to overwrite the existing list or click **Cancel** and specify a new name.
  - Move the vertical cursor line to the desired frequencies.
  - Left-click the desired frequencies.
  - Click to save the frequency list and return.

- To view the frequency list:
  - Enter **edlist f1 freqlist** and click **OK**.
  - It should look, for example, like this:

    ```
    O 300.19
    3759.74
    8775.99
    8712.96
    ```

  - Click **Cancel** to close the list.
- Enter **stdisp** to start the ShapeTool interface.
- Proceed as follows:
  - Click  , choose **Shape**, select *Sinc1.1000* and click **OK**

    or
  - Click  | **Classical Shapes** | **Sinc** and enter:

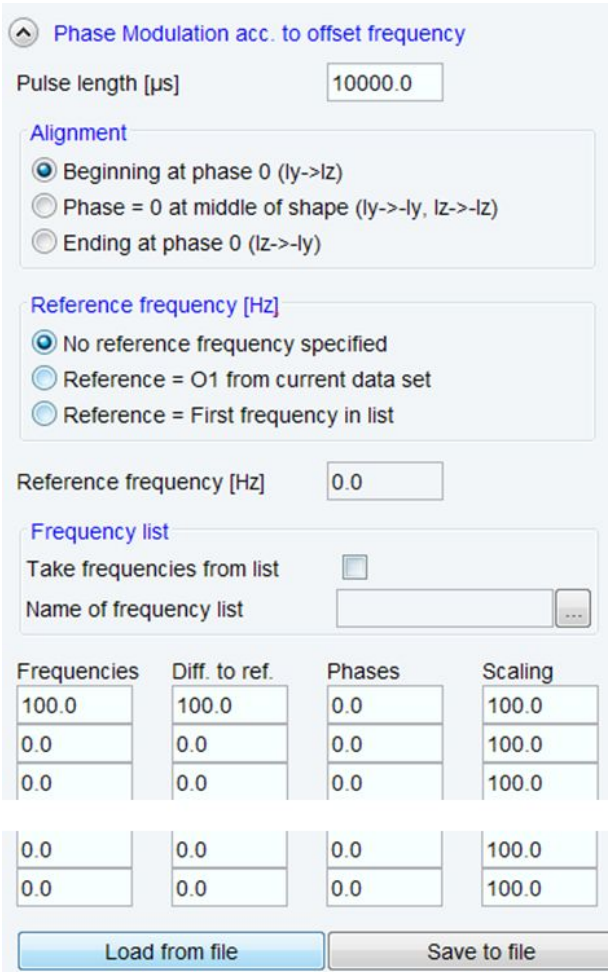
    Size of shape: 1000

    Number of cycles: 1

    in the parameter section.
- Click  | **Multiple Phase Modulation**
  In the appearing dialog box:
  - Select Ending at Phase 0.
  - Select Reference = O1 from current Data Set.
  - Check the box Take Frequencies from list.
  - Set Length of Pulse to 10000.
  - Set Name of Frequency List to freqlist or the name you specified in step 2.
  - Click **OK**.
- Check the frequency values in the appearing dialog box and click **OK**.

Click  and click **Shape,** enter a *Name* and *Title*, set the *Rotation Angle* to *90°* and the *Type of Rotation* to *excitation*

Now you have created a shape that can be used in a WET experiment.



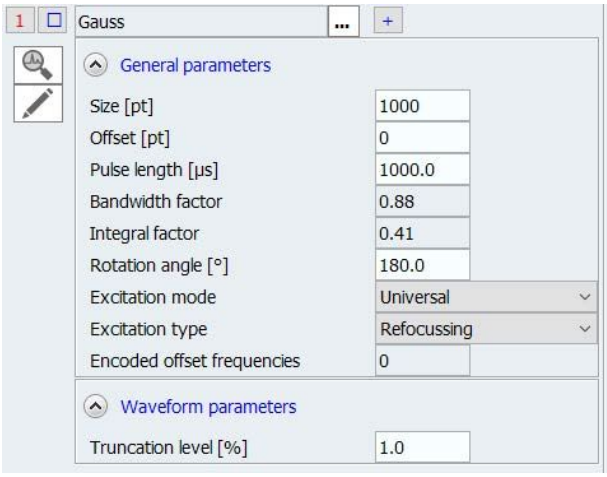
Example 2: Two-step Procedure to a Selective Experiment

- Read/acquire a 1D reference spectrum.
- Determine the width of the region to be excited. Let's assume this is 40 Hz.
- Create a new dataset.
- Read the pulse program *selgpse.*
- Enter **ased** and set all acquisition parameters for *selgpse*, except for the shaped pulse parameters.
- Enter **stdisp** to start the ShapeTool interface.
- Click ... | **Classical Shapes** => **Gauss**
- In the parameter section, enter:

  Size: 1000

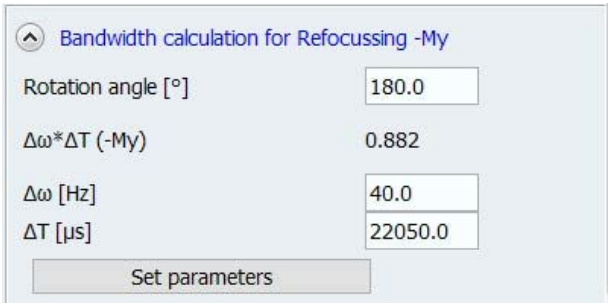
  Excitation type: Refocussing

  Truncation level (%): 1

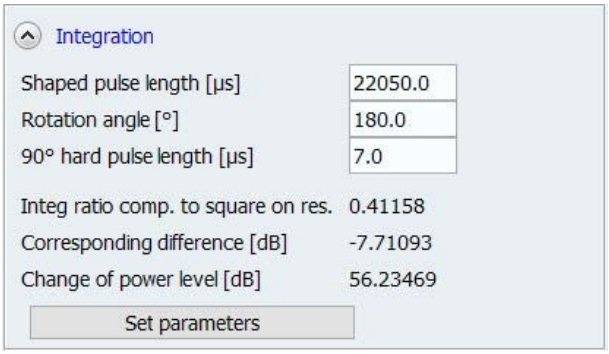


- Click  | **Define ASED Parameters**
  - Set *Length of shaped pulse* to P12
  - Set *Power level of shaped pulse* to SPW2
  - Set *Name of shaped pulse* to SPNAM2 and click **OK**.
- Click  | **Calculate Bandwidth for Refocusing -My**

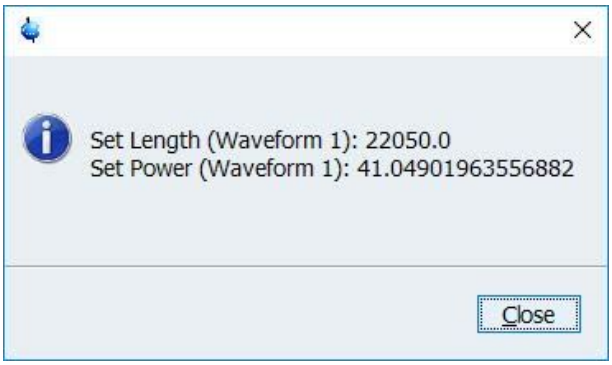
  This allows to calculate the pulse length (DeltaT) from the bandwith factor which is the product of the width of excitation (DeltaOmega) and the pulse length.
  - Enter Delta Omega: 40 (the bandwidth determined previously)



- Click  | **Integrate Shape**

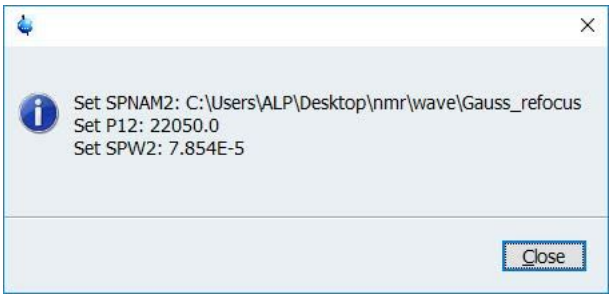


- Click **Set parameters** to calculate the power value (in dBW) which is equal to the sum of the Change of power level and the hard power level corresponding to the 90° hard pulse (default: PLdB1).



- Click  | **Set Parameters to ASED**
- Define the name of the shape file (ex. Gauss_refocus)

- Click OK to store the values of the corresponding parameters (**SPNAM2, P12** and **SPW2**) and the associated dataset.



- Type ased on the associated dataset if you want to check these values.
- If the region to be excited is off-resonance, the parameters SPOFFS2 and SPOAL2 must be set accordingly: you need to determine the chemical shift difference between the peak and O1 and set the parameter SPOFFS2 accordingly. To determine the proper value, click  , put the cursor on the desired peak and subtract the value of SFO1 from the MHz value displayed in the upper left of the data field. Moreover, the value of the parameter SPOAL2 must be set to 0.5 (refocusing pulse).

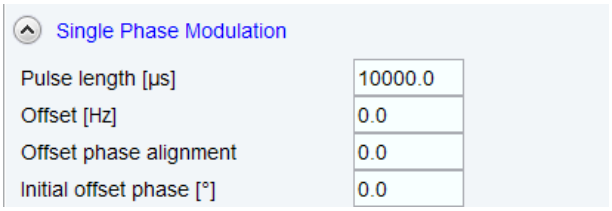
Example 3: Create a Shaped Pulse for Sel. Excitation from Integral Regions

- Read/acquire a 1D reference spectrum.
- Create a new dataset.
- Read the pulse program *selgpse.*
- Enter **ased** and set all acquisition parameters for *selgpse*, except for the shaped pulse parameters.
- Switch back the reference spectrum.
- Enter **stdisp** to start the ShapeTool interface.
- Define excitation region interactively ().

- Click  | **Classical Shapes** => **Gauss**
- In the parameter section, enter:
  Size of shape: 1000
  Truncation level: 1
- Click  | **Define ASED Parameters**
  - Set *Length of shaped pulse* to P12
  - Set *Power level of shaped pulse* to SPW2
  - Set *Name of shaped pulse* to SPNAM2

?????????????

- Click  | **Calc. Shape from excitation Region**
- Click **OK**
- Click **Options** => **Select associated dataset** and select the *selgpse* dataset.
- Click **update parameters**, specify a name for the shaped pulse and click **OK**.
- Now switch to the *selgpse* dataset and start the experiment.



3 Generating Shapes

3.1 Introduction

Generating a shape with the **st** command or with the C function generateShapeC() will produce a shape consisting of one waveform specified by the function name and the parameters given.

Mandatory parameters include the function name and the size of the shape in points. Parameters must be given in the order shown at the function descriptions below. If parameters are omitted at the end of the parameter list, a default value is applied.

To distinguish shape and gradient generation in the **st** command after the size parameter a <false> can be inserted to generate a gradient shape. The generateShapeC() function takes as the third parameter an integer value being either 1 or 0 indicating shape and gradient generation respectively.

By default the st command stores the output file under the name <function>. If you prefer a different filename, this must be specified with the option: <filename=*newname*>. The option <-*nobwcalc*> will prevent automatic calculation of the bandwidth factor resulting in a bandwith factor of 0.0.

General Syntax

st generate <function> <size> [false] [filename=<outfile>] [-nobwcalc] [<parameters>]

generateShapeC(const char* function, const char* params, int phaseData);

Example:

st generate Gauss 1000 false filename=Gauss.new -nobwcalc 1.0

This will generate a gradient with a Gaussian waveform with 1000 points and a truncation level of 1%. The shape is stored as *Gauss.new*. No automatic calculation of the bandwidth factor is done.

generateShapeC(“Gauss”, “1000 1.0”, 1);

This will generate a shape with a Gaussian waveform with 1000 points and a truncation level of 1%.

3.2 Basic Shapes

3.2.1 Rectangle

Syntax

st generate Rectangle <NPOINTS> <AMP=100.0>

generateShapeC(“Rectangle”, “<NPOINTS> <AMP=100.0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double AMP = amplitude in %

3.2.2 Exponentail Function (Efunc)

Syntax

st generate Efunc <NPOINTS> <TRUNCLEV=0.1>

st generate EfuncAlt <NPOINTS> <LB=44.0> <SW=10000.0>

generateShapeC(“Efunc”, “<NPOINTS> <TRUNCLEV=0.1>”, <phaseData>);

generateShapeC(“EfuncAlt”, “<NPOINTS> <LB=44.0> <SW=10000.0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double TRUNCLEV = truncation level [%]

double LB = line broadening

double SW = sweepwidth [Hz]

Remarks

TRUNCLEV = LB / SW = exp(-beta * (NPOINTS-1) *100 with beta = (LB*NPOINTS) / (2*SW)

3.2.3 Ramp

Syntax

st generate Ramp <NPOINTS> <STARTAMP=0.0> <ENDAMP=100.0>

generateShapeC(“Ramp”, “<NPOINTS> <STARTAMP=0.0> <ENDAMP=100.0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double STARTAMP = amplitude in %

double ENDAMP = amplitude in %

3.2.4 Quadratic Ramp (QRamp)

Syntax

st generate QRamp <NPOINTS> <STARTAMP=0.0> <ENDAMP=100.0>

generateShapeC(“QRamp”, “<NPOINTS> <STARTAMP=0.0> <ENDAMP=100.0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double STARTAMP = amplitude in %

double ENDAMP = amplitude in %

3.2.5 Sinus

Syntax

st generate Sinus <NPOINTS> <CYCNUM=3.0> <PHASE=0.0>

generateShapeC(“Sinus”, “<NPOINTS> <CYCNUM=3.0> <PHASE=0.0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

int CYCNUM = number of cycles

double PHASE = phase angle [deg] (0.0 <= p <= 360.0)

3.2.6 Trapezoid

Syntax

st generate Trapezoid <NPOINTS> <STARTAMP=0.0> <LEFTLIMIT=300> <CENTERAMP=100.0> <RIGHTLIMIT=700> <ENDAMP=0.0>

generateShapeC(“Trapezoid”, “<NPOINTS> <STARTAMP=0.0> <LEFTLIMIT=300> <CENTERAMP=100.0> <RIGHTLIMIT=700> <ENDAMP=0.0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double STARTAMP = amplitude at start of shape [%]

int LEFTLIMT = left limit of center region [pt]

double CENTERAMP = amplitude at center of shape [%]

int RIGHTLIMT = right limit of center region [pt]

double ENDAMP = amplitude at end of shape [%]

3.2.7 Triangle

Syntax

st generate Triangle <NPOINTS>

generateShapeC(“Triangle”, “<NPOINTS”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

3.3 Classical Shapes

3.3.1 Burp

BurpType = {EBurp1 | EBurp2 | IBurp1 | IBurp2 | ReBurp | UBurp}

Syntax

st generate <BurpType> <NPOINTS>

generateShapeC("<BurpType>", "<NPOINTS>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

Literature

H. Geen & R. Freeman, J. Magn. Reson. 93, 93-141 (1991).

3.3.2 Gauss

Syntax

st generate Gauss <NPOINTS> <TRUNCLEV=1.0>

generateShapeC("Gauss", "<NPOINTS> <TRUNCLEV=1.0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double TRUNCLEV = truncation level [%]

Literature

C. Bauer, R. Freeman, T. Frenkiel, J. Keeler & A.J. Shaka.

J. Magn. Reson. 58, 442-457 (1984).

L. Emsley & G. Bodenhausen, J. Magn. Reson. 82, 211-221 (1989)

3.3.3 GaussCascades

CascadeType = { GaussCascadeG3 | GaussCascadeG4 |

GaussCascadeQ3 | GaussCascadeQ5 |

GaussCascadeOwn}

Syntax

st generate <CascadeType> <NPOINTS>

generateShapeC("<CascadeType>", "<NPOINTS>", <phaseData>);

st generate GaussCascadeOwn <NPOINTS> <N=1> (<POS=50.0> <AMP=100.0> <WITH=10.0>)*N>

generateShapeC("GaussCascadeOwn", "<NPOINTS> <N (POS AMP WIDTH)*N>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

int N = number of coefficients

double POS = relative position of Gaussian [%]

double AMP = relative amplitude of Gaussian [%]

double WIDTH = relative width of Gaussian [%]

Remarks

GaussCascadeOwn needs as parameters the number of coefficients N followed by N times POS AMP WIDTH.

The maximum number of coefficients is 20.

Literature

L. Emsley & G. Bodenhausen, Chem. Phys. Lett. 165, 469 (1990).

L. Emsley & G. Bodenhausen, J. Magn. Reson. 97, 135-148 (1992).

3.3.4 HalfGauss

Syntax

st generate HalfGauss <NPOINTS> <TRUNCLEV=1.0>

generateShapeC("HalfGauss", "<NPOINTS> <TRUNCLEV=1.0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double TRUNCLEV = truncation level [%]

Literature

J. Friedrich, S. Davies & R. Freeman, J. Magn. Reson. 75, 390-395 (1987).

3.3.5 Hermite

Syntax

st generate Hermite <NPOINTS> <TRUNCLEV=1.0> <QUADCOEFF=1.0>

generateShapeC("Hermite", "<NPOINTS> <TRUNCLEV=1.0> <QUADCOEFF=1.0>", **<**phaseData>);

Parameters

int NPOINTS = shape size [pt]

double TRUNCLEV = truncation level [%]

double QUADCOEFF= coefficient of quadratic term (-4.0 .... +6.0)

Literature

W.S. Warren, J. Chem. Phys. 81, 5437-5448 (1984).

3.3.6 Seduce

Syntax

st generate <SeduceType> <NPOINTS>

generateShapeC("<SeduceType>", "<NPOINTS>", <phaseData>);

Parameters

SeduceType = { Seduce1 | Seduce3 }

int NPOINTS = shape size [pt]

Literature

M.A. McCoy & L. Mueller, J. Magn. Reson. A 101, 122-130 (1993).

3.3.7 Sinc

Syntax

st generate Sinc <NPOINTS> <CYCNUM=3.0>

generateShapeC("Sinc", "<NPOINTS> <CYCNUM=3.0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

int CYCNUM = number of cycles

Literature

A.J. Temps Jr. & C.F. Brewer, J. Magn. Reson. 56, 355-372 (1984).

3.3.8 Sneeze

Syntax

st generate Sneeze <NPOINTS>

generateShapeC("Sneeze", "<NPOINTS>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

Literature

J. M. Nuzillard & R. Freeman, J. Magn. Reson. A 110, 252-256 (1994).

3.3.9 QSneeze

Syntax

st generate QSneeze<NPOINTS>

generateShapeC(“QSneeze”, “<NPOINTS>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

3.3.10 Snob

Syntax

st generate <SnobType> <NPOINTS>

generateShapeC(“<SnobType>”, “<NPOINTS>”, <phaseData>);

Parameters

SnobType = { ESnob | ISnob2 | ISnob3 | RSnob | DSnob }

int NPOINTS = shape size [pt]

Literature

E. Kupce, J. Boyd & I.D. Campbell, J. Magn. Reson. B 106, 300-303 (1995).

3.3.11 Vega Shapes

Syntax

st generate <VegaType> <NPOINTS>

generateShapeC(“<VegaType>”, “<NPOINTS>”, <phaseData>);

Parameters

VegaType = { EVega1 | EVega2 | IVega }

int NPOINTS = shape size [pt]

Literature

D. Abramovich & S. Vega, J. Magn. Reson. A 105, 30-48 (1993).

3.3.12 Shapes defined by Fourier Coefficients (ShapFour)

Syntax

st generate ShapFour <NPOINTS> <N=1> <A0=1.0> (<A=1.0>)*N (<B=1.0>)*N

generateShapeC(“ShapFour”, “<NPOINTS> <N=1> <A0=1.0> (<A=1.0>)*N (<B=1.0>)*N”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

int N = number of coefficients

double <A0> = a0-coefficient

double <A1> … <AN> = a-coefficients

double <B1> … <BN> = b-coefficients

Remarks

ShapeFour needs as parameters the number of coefficients N followed by A0 followed by N times <Ax> followed by N times <Bx>

The maximum number of coefficients is 100.

3.4 Adiabatic Shapes

3.4.1 Hyperbolic Secant Shape (HypSec)

Syntax

st generate HypSec <NPOINTS> <SW=20.0> <TRUNCLEV=1.0> <FULLHALF=true> <SWDIR=-1>

generateShapeC("HypSec", "<NPOINTS> <SW=20.0> <TRUNCLEV=1.0> <FULLHALF=true> <SWDIR=-1>", <phaseData>);

st generate HypSecAlt <NPOINTS> <MU=5.929444> <BETA=5.298292> <FULLHALF=true> <SWDIR=-1>

generateShapeC("HypSecAlt", "<NPOINTS> <MU=5.929444> <BETA=5.298292> <FULLHALF=true> <SWDIR=-1>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double MU = mu value

double BETA = beta value

double TRUNCLEV = truncation level [%]

boolean FULLHALF = full or half passage [true / false]

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

M.S. Silver, R.I. Joseph & D.I. Hoult, J. Magn. Reson. 59, 347 (1984).

3.4.2 SinCos

Syntax

st generate SinCos <NPOINTS> <PHASEFAC=4.0> <FULLHALF=true> <SWDIR=-1>

generateShapeC(“SinCos”, “<NPOINTS> <PHASEFAC=4.0> <FULLHALF=true> <SWDIR=-1>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double PHASEFAC = phase amplitude factor (0.0 <= p <= 16.0)

boolean FULLHALF = full or half passage [true / false]

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

M.R. Bendall & D.T. Pegg, J. Magn. Reson. 67, 376-381 (1986).

3.4.3 SmoothedChirp Shape

Syntax

st generate SmoothedChirp <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <SMOOTHED=20.0> <SWDIR=-1> <SMOOTHING=0>

generateShapeC(“SmoothedChirp”, “<NPOINTS> <SW=40000.0> <LENGTH=1500.0> <SMOOTHED=20.0> <SWDIR=-1> <SMOOTHING=0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double SMOOTHED = % to be smoothed

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

int SMOOTHING = smoothing (-pi/2 -> +pi/2) 0: false, 1:true

Literature

J. M. Boehlen & G. Bodenhausen, J. Magn. Reson. A 102, 293 (1993).

3.4.4 Composite SmoothedChirp Shape

Syntax

st generate CompositeSmoothedChirp <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <SMOOTHED=20.0> <SWDIR=-1> <SMOOTHING=0>

generateShapeC(“CompositeSmoothedChirp”, “<NPOINTS> <SW> <SW=40000.0> <LENGTH=1500.0> <SMOOTHED=20.0> <SWDIR=-1> <SMOOTHING=0>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double SMOOTHED = % to be smoothed

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

int SMOOTHING = smoothing (-pi/2 -> +pi/2) 0: false, 1:true

Literature

T.L. Hwang, P.C.M van Zijl & M. Garwood J. Magn. Reson. 124, 250 (1997).

3.4.5 TanhTan Shape

Syntax

st generate TanhTan <NPOINTS> <SW=1000000.0> <LENGTH=500.0> <ZETA=10.0> <TANKAPPA=20.0> <SWDIR=-1>

generateShapeC("TanhTan", "<NPOINTS> <SW=1000000.0> <LENGTH=500.0> <ZETA=10.0> <TANKAPPA=20.0> <SWDIR=-1>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double ZETA = Value for zeta

double TANKAPPA = Value for tan(kappa)

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

R.S. Staewen, A.J. Johnson, B.D. Ross, T. Parrish, H. Merkle & M. Gar- wood,

Invest. Radiol. 25, 559-567(1990) M. Garwood & Y. Ke,

J. Magn. Reson. 94 511-525(1991).

3.4.6 Wurst Shape

Syntax

st generate Wurst <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <POWERINDEX=20.0> <SWDIR=-1>

generateShapeC("Wurst", "<NPOINTS> <SW=40000.0> <LENGTH=1500.0> <POWERINDEX=20.0> <SWDIR=-1>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double POWERINDEX = Amplitude Power Index

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

E. Kupce & R. Freeman, J. Magn. Reson. A 115, 273-276 (1995).

3.5 Constant-Adiabaticity Shapes

3.5.1 CaPowHsec

Syntax

st generate CaPowHsec <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <BETA=5.3> <POWERINDEX=8.0> <SWDIR=-1>

generateShapeC("CaPowHsec", "<NPOINTS> <SW=40000.0> <LENGTH=1500.0> <BETA=5.3> <POWERINDEX=8.0> <SWDIR=-1>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double BETA = beta value

double POWERINDEX = Amplitude Power Index

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

A. Tannus & M. Garwood, J. Magn. Reson. A 120, 133-137 (1996).

3.5.2 CaSmoothedChirp

Syntax

st generate CaSmoothedChirp <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <SMOOTHED=20.0> <SWDIR=-1> <SMOOTHING=0>

generateShapeC("CaSmoothedChirp", "<NPOINTS> <SW=40000.0> <LENGTH=1500.0> <SMOOTHED=20.0> <SWDIR=-1> <SMOOTHING=0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double SMOOTHED = % to be smoothed

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

int SMOOTHING = smoothing (-pi/2 -> +pi/2) 0: false, 1:true

Literature

A. Tannus & M. Garwood, J. Magn. Reson. A 120, 133-137 (1996).

3.5.3 CaGauss

Syntax

st generate CaGauss <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <TRUNCLEV=1.0> <SWDIR=-1>

generateShapeC(“CaGauss”, “<NPOINTS> <SW=40000.0> <LENGTH=1500.0> <TRUNCLEV=1.0> <SWDIR=-1>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double TRUNCLEV = truncation level [%]

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

A. Tannus & M. Garwood, J. Magn. Reson. A 120, 133-137 (1996).

3.5.4 CaLorentz

Syntax

st generate CaLorentz <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <TRUNCLEV=1.0> <SWDIR=-1>

generateShapeC(“CaLorentz”, “<NPOINTS> <SW=40000.0> <LENGTH=1500.0> <TRUNCLEV=1.0> <SWDIR=-1>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double TRUNCLEV = truncation level [%]

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

A. Tannus & M. Garwood, J. Magn. Reson. A 120, 133-137 (1996).

3.5.5 CaWurst

Syntax

st generate CaWurst <NPOINTS> <SW=40000.0> <LENGTH=1500.0> <POWERINDEX=2.0> <SWDIR=-1>

generateShapeC(“CaWurst”, “<NPOINTS> <SW=40000.0> <LENGTH=1500.0> <POWERINDEX=2.0> <SWDIR=-1>”, <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double SW = sweepwidth [Hz]

double LENGTH = duration [us]

double POWERINDEX = Amplitude Power Index

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Literature

A. Tannus & M. Garwood, J. Magn. Reson. A 120, 133-137 (1996).

3.6 Special Shapes for Solids Applications

3.6.1 Sines

Syntax

st generate Sines <NPOINTS> <CYCNUM=2.0> <PHASE=0.0> <AVERAGE=80.0> <MODAMP=10.0>

generateShapeC("Sines", "<NPOINTS> <CYCNUM=2.0> <PHASE=0.0> <AVERAGE=80.0> <MODAMP=10.0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double CYCNUM = number of cycles

double PHASE = initial phase [deg]

double AVERAGE = average amplitude

double MODAMP = amplitude of modulation

Literature

S.Hediger, B.H.Meier, R.H. Ernst, J. Chem. Phys. 102, 4000-4011 (1995).

3.6.2 TangAmplitudeMod Shape

Syntax

st generate TangAmplitudeMod <NPOINTS> <MODAMP=5000.0> <PHASE=26.565051> <AVERAGE=50000.0> <AMPFAC=50.0>

generateShapeC("TangAmplitudeMod", "<NPOINTS> <MODAMP=5000.0> <PHASE=26.565051> <AVERAGE=50000.0> <AMPFAC=50.0>", <phaseData>);

st generate TangAmplitudeModAlt <NPOINTS> <MODAMP=5000.0> <DIPOLCPL=10000.0> <AVERAGE=50000.0> <AMPFAC=50.0>

generateShapeC("TangAmplitudeModAlt", "<NPOINTS> <MODAMP=5000.0> <DIPOLCPL=10000.0> <AVERAGE=50000.0> <AMPFAC=50.0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double MODAMP = Amplitude of Modulation

double DIPOLCPL = Dipolar Coupling

double PHASE = Phase Value (max. 95 degree)

double AVERAGE = Average Amplitude

double AMPFAC = Amplitude Scaling Factor

Literature

M. Baldus, D.C. Geurts, S.Hediger, B.H.Meier,
J. Magn. Reson. A 118, 140-144 (1996).

3.7 Special Shapes for Imaging Applications

3.7.1 CosSinc Shapes

Syntax

st generate CosSinc <NPOINTS> <CYCNUM=3.0> <WINDOWSIZE=50.0>

generateShapeC("CosSinc", "<NPOINTS> <CYCNUM=3.0> <WINDOWSIZE=50.0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

int CYCNUM = number of cycles

double WINDOWSIZE = window size in %

Literature

G.J. Galloway, W.M. Brooks, J.M. Bulsing, I.M. Brereton,
J. Field, M. Irving, H. Baddeley & D.M. Doddrell, J. Magn. Reson. 73, 360-368 (1987).

3.7.2 Sine * Sinc Shape (SineSinc)

Syntax

st generate SinSinc <NPOINTS> <CYCNUM=7.0> <PHASEFAC=3.0> < PHASE=0.0>

generateShapeC("SinSinc", "<NPOINTS> <CYCNUM=7.0> <PHASEFAC=3.0> <PHASE=0.0>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

double CYCNUM = number of cycles

double PHASEFAC = Factor for Sine Argument

double PHASE = Initial Phase for Sine [deg]

Literature

D.M. Doddrell, J.M. Bulsing, G.J. Galloway, W.M. Brooks, J. Field, M. Irving, & H. Baddeley, J. Magn. Reson. 70, 319-326 (1986).

3.8 Special Shapes for Decoupling

3.8.1 Swirl Shapes

SwirlType = { Swirl11 | Swirl12 | Swirl17 }

Syntax

st generate <SwirlType> <NPOINTS>

generateShapeC("<SwirlType>", "<NPOINTS>", <phaseData>);

Parameters

int NPOINTS = shape size [pt]

Literature

H. Geen & J.-M. Boehlen, J. Magn. Reson. 125, 376-382 (1997).

4 Manipulating Shapes

4.1 Introduction

Next to the possibility of manipulating shapes with the interactive ShapeTool, these operations can be done by either using the command line tool **st** or invoking the corresponding function from within an AU program.

The following manipulating commands are implemented:

soffs – single phase modulation according to offset frequencies.

offs – multiple phase modulation according to offset frequencies.

sinm2 - single sine modulation (+/- offset).

cosm2 - single cosine modulation (+/- offset).

sweep - modulation according to linear frequency sweep-

caSweep - modulation according to constant adiabatic frequency sweep.

power - calculate power of amplitude.

scale - scale the amplitude of a shape.

addphase – add a constant phase.

trev - time reverse a shape.

expand - expand shape according to supercycle.

General Syntax

st manipulate <filename> <command> <parameters> [filename=<outfile>]

const double* manipulateShapeC(const char* fileName, const char* command, const char* params, const char* freqList, const double** phasePtr, int* nPoints, int* shapeMode);

Examples

st manipluate Gauss offs b 1000 1 100

This will impose a frequency offset of 100 Hz to the shape contained in the file “Gauss” with a length of 1000 µs and phase alignment at the beginning of the shape.

ampPtr = manipulateShapeC(“Gauss”, “offs b”, “1000 1 100”, 0, phasePtr, nPoints, shapeMode);

Same as above but the manipulated file is not saved to disc but instead the shape data is returned through the pointers ampPtr and *phasePtr, information about the new size and mode (# of offset frequencies) is returned through nPoints and shapeMode.

4.2 Modulation according to Frequency Offset

4.2.1 Command soffs

Calculates a phase modulation according to a single offset frequency.

Syntax

st manipulate <filename> soffs <LENGTH=10000.0> <OFFSET=0.0> [<ALIGN=0.0>] [<PHASE=0.0>]

ampPtr = manipulateShapeC(fileName, “soffs”, “<LENGTH=10000.0> <OFFSET=0.0> [<ALIGN=0.0>] [<PHASE=0.0>]”, 0, phasePtr, nPoints, shapeMode);

Parameters

double LENGTH = length of shaped pulse [us]

double OFFSET = nth offset frequency [Hz]

double ALIGN = nth alignment factor [0.0 – 1.0]

double PHASE = nth initial phase [deg]

Example

st manipulate Gauss soffs 1000 100 0.0 90.0

ampPtr = manipulateShapeC(fileName, “soffs”, “1000 100 0.0 90.0”,

0, phasePtr, nPoints, shapeMode);

4.2.2 Command offs

Calculates a phase modulation according to multiple offset frequencies. There are several optional modifiers to specify the operation:

b - phase modulation beginning at phase 0

m - phase modulation with phase of 0 at middle of shape

e - phase modulation ending at phase 0

f - frequencies taken from frequency list

p - additional phase setting

s - additional scaling

Syntax

st manipulate <filename> offs [b|m|e] [f, p, s] <LENGTH> <N> (<OFFSET> [<PHASE>] [<SCALE>])*N [<freqlist>]

ampPtr = manipulateShapeC(fileName, “offs [b|m|e] [f, p, s]”, “<LENGTH> <N> (<OFFSET> [<PHASE>] [<SCALE>])*N”,

freqList, phasePtr, nPoints, shapeMode);

Parameters

double LENGTH = length of shaped pulse [us]

int N = number of offset frequencies

double OFFSET = nth offset frequency [Hz]

double SCALE = nth scaling factor [%]

double PHASE = nth initial phase [deg]

string freqlist = file with frequency list

Example

st manipulate Gauss offs b 1000 2 2000 3000

ampPtr = manipulateShapeC(“Gauss”, “offs b”, “1000 2 2000 3000”,

0, phasePtr, nPoints, shapeMode);

st manipulate Gauss offs b f 1000 freqlist

ampPtr = manipulateShapeC(“Gauss”, “offs b f”, “1000”,

freqList, phasePtr, nPoints, shapeMode);

st manipulate Gauss offs m p 1000 2 2000 90 3000 300

ampPtr = manipulateShapeC(“Gauss”, “offs m p”, “1000 2 2000 90 3000 300”, 0, phasePtr, nPoints, shapeMode);

st manipulate Gauss offs e s 1000 2 2000 50 3000 75

ampPtr = manipulateShapeC(“Gauss”, “offs e s”, “1000 2 2000 50 3000 75”, 0, phasePtr, nPoints, shapeMode);

st manipulate Gauss offs m f s 1000 50 75 freqlist

ampPtr = manipulateShapeC(“Gauss”, “offs m f s”, “1000 50 75”,

freqList, phasePtr, nPoints, shapeMode);

st manipulate Gauss offs e f p 1000 90 180 freqlist

ampPtr = manipulateShapeC(“Gauss”, “offs e f p”, “1000 90 180”,

freqList, phasePtr, nPoints, shapeMode);

4.3 Amplitude Modulation

4.3.1 Command sinm2

Calculates an amplitude modulation such that the pulse excites at two symmetric sidebands (+/- offset) with opposite phase.

Syntax

st manipulate <filename> sinm2 <LENGTH=10000.0> <OFFFREQ=3000.0>

ampPtr = manipulateShapeC(fileName, “sinm2”, “<LENGTH=10000.0> <OFFFREQ=3000.0>”,0, phasePtr, nPoints, shapeMode);

Parameters

double LENGTH = length of shaped pulse [us]

double OFFFREQ = offset frequency [Hz]

Example

st manipulate Gauss sinm2 1000 3000

ampPtr = manipulateShapeC(“Gauss”, “sinm2”, “1000 3000”,

0, phasePtr, nPoints, shapeMode);

4.3.2 Command cosm2

Calculates an amplitude modulation such that the pulse excites at two symmetric sidebands (+/- offset) with the same phase.

Syntax

st manipulate <filename> cosm2 <LENGTH=10000.0> <OFFFREQ=3000.0>

ampPtr = manipulateShapeC(fileName, “cosm2”, “<LENGTH=10000.0> <OFFFREQ=30000.0>”,0, phasePtr, nPoints, shapeMode);

Parameters

double LENGTH = length of shaped pulse [us]

double OFFFREQ = offset frequency [Hz]

Example

st manipulate Gauss cosm2 1000 3000

ampPtr = manipulateShapeC(“Gauss”, “cosm2”, “1000 3000”,

0, phasePtr, nPoints, shapeMode);

4.4 Modulation according to Frequency Sweep

4.4.1 Command sweep

Calculates a phase modulation according to a linear frequency sweep.

Syntax

st manipulate <filename> sweep

<LENGTH=10000.0><SW=40000.0><SWDIR=-1>

ampPtr = manipulateShapeC(fileName, “sweep”, “<LENGTH=10000.0> <SW=40000.0> <SWIR=-1>”,0, phasePtr, nPoints, shapeMode);

Parameters

double LENGTH = length of shaped pulse [us]

double SW = total sweep-width [Hz]

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Example

st manipulate Gauss sweep 1000 5000

ampPtr = manipulateShapeC(“Gauss”, “sweep”, “1000 5000”,

0, phasePtr, nPoints, shapeMode);

4.4.2 Command casweep

Calculates a phase modulation according to a constant adiabaticity frequency sweep.

Syntax

st manipulate <filename> casweep <LENGTH=10000.0><SW=40000.0><SWDIR=-1>

ampPtr = manipulateShapeC(fileName, “casweep”, “<LENGTH=10000.0> <SW=40000.0> <SWIR=-1>”, 0, phasePtr, nPoints, shapeMode);

Parameters

double LENGTH = length of shaped pulse [us]

double SW = total sweep-width [Hz]

int SWDIR = Direction of sweep: 1= high to low, -1 = low to high field.

Example

st manipulate Gauss casweep 1000 5000

ampPtr = manipulateShapeC(“Gauss”, “casweep”, “1000 5000”,

0, phasePtr, nPoints, shapeMode);

4.5 Command power

Calculate the power of amplitudes. The data is scaled to maximum amplitude.

Syntax

st manipulate <filename> power <EXPONENT=2.0>

ampPtr = manipulateShapeC(fileName, “power”, “<exponent=2.0>”, 0, phasePtr, nPoints, shapeMode);

Parameters

double EXPONENT = exponential factor

Example

st manipulate Gauss power 2.5

ampPtr = manipulateShapeC(“Gauss”, “power”, “2.5”,

0, phasePtr, nPoints, shapeMode);

4.6 Command scale

Scale the amplitude of a shape to a given percentage.

Syntax

st manipulate <filename> scale <SCALE=100.0>

ampPtr = manipulateShapeC(fileName, “scale”, “<SCALE=100.0>”, 0, phasePtr, nPoints, shapeMode);

Parameters

double SCALE = new scaling in %

Example

st manipulate Gauss scale 50.0

ampPtr = manipulateShapeC(“Gauss”, “scale”, “50.0”, 0, phasePtr, nPoints, shapeMode);

4.7 Command addphase

Adds a constant phase to a shape.

Syntax

st manipulate <filename> addphase <PHASE=90.0>

ampPtr = manipulateShapeC(fileName, “addphase”, “<PHASE=90.0>”, 0, phasePtr, nPoints, shapeMode);

Parameters

double PHASE = phase constant to be added [deg]

Example

st manipulate Gauss 90.0

ampPtr = manipulateShapeC(“Gauss”, “addphase”, “90.0”, 0, phasePtr, nPoints, shapeMode);

4.8 Command trev

Time reverse a shape. Since shapes are often optimized for a particular rotation (e.g. Iz ’ - Iy , excitation) , reversing the rotation (Iy ’ Iz , flipback) will only work reliably if the shape is time reversed as well.

Syntax

st manipulate <filename> trev

ampPtr = manipulateShapeC(fileName, “trev”, “”, 0, phasePtr, nPoints, shapeMode);

Parameters

trev needs no additional parameters.

Example

st manipulate HalfGauss trev

ampPtr = manipulateShapeC(“HalfGauss”, “trev”, “”, 0, phasePtr, nPoints, shapeMode);

4.9 Command expand

Expand shape according to a phase list (supercycle).

Syntax

st manipulate <filename> expand <N=4> <(PHASE)*N = 0.0 0.0 180.0 180.0>

ampPtr = manipulateShapeC(fileName, “expand”, “<N=4> <(PHASE)*N = 0.0 0.0 180.0 180.0>”, 0, phasePtr, nPoints, shapeMode);

Parameters

int N= number of phases to expand

double PHASE = nth phase value [deg]

Example

st manipulate Gauss expand 4 0 0 180 180

ampPtr = manipulateShapeC(“Gauss”, “expand”, “4 0.0 0.0 180.0 180.0”, 0, phasePtr, nPoints, shapeMode);

4.10 Command fraction

Cuts out part of shape.

Syntax

st manipulate <filename> fraction <FRACSTART=0> <FRACEND= NPOINTS>

ampPtr = manipulateShapeC(fileName, “fraction”, “<FRACSTART=0> <FRACEND=NPOINTS>”, 0, phasePtr, nPoints, shapeMode);

Parameters

int FRACSTART= start point of fraction

int FRACEND= end point of fraction

Example

st manipulate Gauss fraction 200 800

ampPtr = manipulateShapeC(“Gauss”, “fraction”, “200 800”, 0, phasePtr, nPoints, shapeMode);

5 Analyzing Shapes

5.1 Introduction

Next to the possibility of analyzing shapes with the interactive ShapeTool, these operations can be done by either using the command line tool **st** or invoking the corresponding function from within an AU program.

The following analyzing commands are available:

bandw2 - calculate bandwidth for excitation (√(Mx2+My2)

bandw2i - calculate bandwidth for inversion (-Mz)

bandw2ry - calculate bandwidth for refocusing (-My)

bandw2e - calculate bandwidth for excitation (-My)

bandw2r - calculate bandwidth for refocusing (My'√(Mx2+My2))

bandw2rx - calculate bandwidth for refocusing (+Mx)

integr - integrate shape and compare to square.

integr2 - calculate power level compare to hard pulse.

integr3 - integrate shape and calculate power level compared to hard pulse

calcb1m calculate γB1(max)/π on resonance.

calcb1mo calculate γB1(max)/2π at offset

calcb1adia calculate γB1(max)/2π for adiabatic shapes

integradia - integrate adiabatic shapes and calculate power level compared to hard pulse.

bsiegert3 - calculate Bloch-Siegert shift

calcpav - calculate average power level

General Syntax

st analyze <filename> <command> <parameters>

int analyzeShapeC(const char* fileName, const char* command, const char* params, char* result);

The AU program function analyzeShapeC() returns the number of results and the char* result which holds the comma separated result string.

If no default parameters are specified, the values from the shape are taken.

5.2 Calculating pulse bandwidth

5.2.1 Command bandw2

Calculate the bandwidth for excitation (√(Mx2+My2).

Syntax

st analyze <filename> bandw2 <TOTROT>
numRes = analyzeShapeC(<filename>, “bandw2”, “<TOTROT>”, result);

Parameters

double TOTROT = total rotation [deg]

Results

double BWFAC = bandwidth factor Δω * ΔT.

Example

st analyze Gauss bandw2 90
numRes = analyzeShapeC(“Gauss”, “bandw2”, “90.0”, result);

5.2.2 Command bandw2i

Calculates the bandwidth for inversion (-Mz).

Syntax

st analyze <filename> bandw2i <TOTROT>
numRes = analyzeShapeC(<filename>, “bandw2i”, “<TOTROT>”, result);

Parameters

double TOTROT = total rotation [deg]

Results

double BWFAC = bandwidth factor Δω * ΔT.

Example

st analyze Gauss bandw2i 180
numRes = analyzeShapeC(“Gauss”, “bandw2i”, “180.0”, result);

5.2.3 Command bandw2ry

Calculates bandwidth for refocusing (-My).

Syntax

st analyze <filename> bandw2ry <TOTROT>
numRes = analyzeShapeC(<filename>, “bandw2ry”, “<TOTROT>”, result);

Parameters

double TOTROT = total rotation [deg]

Results

double BWFAC = bandwidth factor Δω * ΔT.

Example

st analyze Gauss bandw2ry 180

numRes = analyzeShapeC(“Gauss”, “bandw2ry”, “180.0”, result);

5.2.4 Command bandw2e

Calculate bandwidth for excitation (-My)

Syntax

st analyze <filename> bandw2e <TOTROT>

numRes = analyzeShapeC(<filename>, “bandw2e”, “<TOTROT>”, result);

Parameters

double TOTROT = total rotation [deg]

Results

double BWFAC = bandwidth factor Δω * ΔT.

Example

st analyze Gauss bandw2e 90

numRes = analyzeShapeC(“Gauss”, “bandw2e”, “90.0”, result);

5.2.5 Command bandw2r

Calculate bandwidth for refocusing (My ’ √(Mx2+My2)

Syntax

st analyze <filename> bandw2r <TOTROT>

numRes = analyzeShapeC(<filename>, “bandw2r”, “<TOTROT>”, result);

Parameters

double TOTROT = total rotation [deg]

Results

double BWFAC = bandwidth factor Δω * ΔT.

Example

st analyze Gauss bandw2r 90

numRes = analyzeShapeC(“Gauss”, “bandw2r”, “90.0”, result);

5.2.6 Command bandw2rx

Calculate bandwidth for refocusing (+Mx)

Syntax

st analyze <filename> bandw2rx <TOTROT>

numRes = analyzeShapeC(<filename>, “bandw2rx”, “<TOTROT>”, result);

Parameters

double TOTROT = total rotation [deg]

Results

double BWFAC = bandwidth factor Δω * ΔT.

Example

st analyze Gauss bandw2rx 90

numRes = analyzeShapeC(“Gauss”, “bandw2rx”, “90.0”, result);

5.3 Calculating integral factors

5.3.1 Command integr

Integrate shape and compare to square pulse.

The command **integr** is not available in the interactive ShapeTool and have been replaced by **integr3**.

Syntax

st analyze <filename> integr

numRes = analyzeShapeC(<filename>, “integr”, “”, result);

Result

double INTEGRATIO= amplitude relative to a square pulse of 100%

double LEVOUT= corresponding difference in dB

Example

st analyze Gauss integr

numRes = analyzeShapeC(“Gauss”, “integr”, “”, result);

5.3.2 Command integr2

Calculates the power level compared to a hard pulse.

The command **integr2** is not available in the interactive ShapeTool and have been replaced by **integr3**.

Syntax

st analyze <filename> integr2 <LENGTH> <HARDPULSE=50.0>

numRes = analyzeShapeC(<filename>, “integr2”, “<LENGTH> <HARDPULSE=50.0>”, result);

Parameters

double LENGTH = length of shaped pulse [us]

double HARDPULSE = length of reference hard pulse [us] of same flip angle

Example

st analyze Gauss integr2 10000.0 50.0

numRes = analyzeShapeC(“Gauss”, “integr2”, “10000.0 50.0”, result);

Result

double INTEGRATIO= amplitude relative to a square pulse of 100%

double LEVOUT= corresponding difference in dB

5.3.3 Command integr3

Integrates shape and calculates power level compared to hard pulse.

Syntax

st analyze <filename> integr3 <LENGTH> <TOTROT> <HARDPULSE=50.0>

numRes = analyzeShapeC(<filename>, “integr3”, “<LENGTH> <TOTROT> <HARDPULSE=50.0>”, result);

Parameters

double LENGTH = length of shaped pulse [us]

double TOTROT = total rotation [deg]

double HARDPULSE = length of reference hard pulse [us] of same flip angle

Example

st analyze Gauss integr3 10000 180 6.6

numRes = analyzeShapeC(“Gauss”, “integr3”, “10000.0 180.0 6.6”, result);

Result

double INTEGRATIO= amplitude relative to a square pulse of 100%

double LEVOUT= corresponding difference in dB

double DB= change of power level compared to level of hard pulse in dB

5.4 Calculating B1(max)

5.4.1 Command calcb1m

Calculates γB1(max)/2π on resonance.

The command **calcb1m** is not available in the interactive ShapeTool and have been replaced by **calcb1mo** with offset = 0.

Syntax

st analyze <filename> calcb1m <LENGTH> <TOTROT>

numRes = analyzeShapeC(<filename>, “calcb1m”, “<LENGTH> <TOTROT>”, result);

Parameters

double LENGTH = length of shaped pulse [us]

double TOTROT = total rotation [deg]

Result

double GB1MAX= maximum γB1 (on resonance)

double REFPUL= corresponding 90 degree square pulse

Example

st analyze Gauss calcb1m 100.0 180.0

numRes = analyzeShapeC(“Gauss”, “calcb1m”, “100.0 180.0”, result);

5.4.2 Command calcb1mo

Calculates γB1(max)/π at offset.

Syntax

st analyze <filename> calcb1mo <LENGTH> <TOTROT><OFFSET>

numRes = analyzeShapeC(<filename>, “calcb1mo”, “<LENGTH> <TOTROT> <OFFSET>”, result);

Parameters

double LENGTH = length of shaped pulse [us]

double TOTROT = total rotation [deg]

double OFFSET = offset [Hz]

Result

double GB1MAX= maximum γB1 (on resonance)

double REFPUL= corresponding 90 degree square pulse

Example

st analyze Gauss calcb1mo 100.0 90.0 3000.0

numRes = analyzeShapeC(“Gauss”, “calcb1mo”, “100.0 90.0 3000.0”, result);

5.4.3 Command calcb1adia

Calculates γB1(max)/2π/√(Q) for adiabatic shapes.

Syntax

st analyze <filename> calcb1adia <LENGTH>

numRes = analyzeShapeC(<filename>, “calcb1adia”, “<LENGTH>”, result);

Parameters

double LENGTH = length of shaped pulse [us]

Result

double SWEEPRATE= sweep rate on resonance [Hz/s]

double FIELDQ= maximum γB1/2π/√(Q) (on resonance)

Example

st analyze SmoothedChirp calcb1adia 10000.0

numRes = analyzeShapeC(“SmoothedChirp”, “calcb1adia”, “10000.0”, result);

5.4.4 Command integradia

Calculates γB1(max)/2π/√(Q) for adiabatic shapes.

Syntax

st analyze <filename> integradia <LENGTH>

numRes = analyzeShapeC(<filename>, “integradia”, “<LENGTH>”, result);

Parameters

double LENGTH = length of shaped pulse [us]

Result

double SWEEPRATE= sweep rate on resonance [Hz/s]

double FIELDQ= maximum γB1/2π/√(Q) (on resonance)

Example

st analyze SmoothedChirp integradia 10000.0

numRes = analyzeShapeC(“SmoothedChirp”, “integradia”, “10000.0”, result);

5.4.5 Command bsiegert3

Calculates the Bloch-Siegert shift.

Syntax

st analyze <filename> bsiegert3 <LENGTH> <GB1MAX> <OFFSET>

numRes = analyzeShapeC(<filename>, “bsiegert”, “<LENGTH> <GB1Max> <OFFSET>”, result);

Parameters

double LENGTH = length of shaped pulse [us]

double GB1MAX = γB1(max) [Hz]

double OFFSET = offset relative to frequency of RF-pulse [Hz]

Result

double DIFFPHASE= phase difference due to Bloch-Siegert shift [deg]

double DIFFFREQ= corresponding frequency shift [Hz]

Example

st analyze Gauss bsiegert3 1000.0 400.0 3000.0

numRes = analyzeShapeC(“Gauss”, “bsiegert”, “1000.0 400.0 3000.0”, result);

5.4.6 Command calcpav

Calculates the average power.

Syntax

st analyze <filename> calcpav

numRes = analyzeShapeC(<filename>, “calcpav”, “”, result);

Example

st analyze Gauss calcpav

numRes = analyzeShapeC(“Gauss”, “calcpav”, “”, result);

Result

double INTEGRATIO= amplitude relative to a square pulse of 100%

double LEVOUT= corresponding difference in dB

double PRATIO= power relative to a square pulse of 100%

6 Miscellaneous st Commands

6.1 Introduction

In addition to generating, manipulating and analyzing shapes, there are additional operations you can invoke from the command line:

Add multiple shapes:

st add <alignment> <n> (<infile> <scale>)*n <outfile>

Merge shapes:

st merge < infile1> <infile2> <outfile>

Convert shapes from ASCII to JCAMP-DX format:

st convert <infile> <outfile> <shapeType> <#freq> [<deltaOmega*deltaT>]

Convert gradients from ASCII to JCAMP-DX format:

st convertgrad <infile> <outfile>

6.2 Add

Add existing shapes to form a new shape.

Syntax

st add <alignment> <n> (<infile> <scale>)*n <outfile>

Parameters

int alignment = with respect to beginning (0), center (1), end (2) of shape

int n = number of shapes to be added

string infile= filename of nth shape to be added

double scale = scaling of nth shape

string outfile = file name of resulting shape

Example

st add 0 2 Gauss.1 100 Gauss.2 50 Gauss.result

6.3 Merge

Merge existing shapes to form a new shape.

Syntax

st merge <infile1> <infile2> <outfile>

Parameters

string infile1= filename of first shape to be merged

string infile2= filename of second shape to be merged

string outfile = file name of resulting shape

Example

st merge Gauss.1 Gauss.2 Gauss.result

6.4 convert, convertgr

This commands convert shapes/gradients from ASCII format to JCAMP-DX format respectively.

Syntax

st convert <infile> <outfile> <EXMODE> <MODE> [<BWFAC>]

st convertgrad <infile> <outfile>

Parameters

string infile =name of file to be converted

string outfile = name of converted file

string EXMODE = Excitation mode of shape
= { Excitation | Inversion | Refocussing |
Universal | Universal180 | Decoupling |
Adiabatic | CompositeAdiabatic | Bib}

int MODE = number of off resonance frequencies encoded in the shape

double BWFAC = deltaOmega*deltaT

Examples

st convert Gauss.ascii Gauss.jcamp Excitation 0

st convertgr Gauss.ascii Gauss.jcamp

7 Shapes in AU programs

7.1 Using ShapeTool commands in AU Programs

Here you find an example of an AU program using the ShapeTool header file ShapeIOC.h.

In this example the command **analyzeShapeC(...)** reads a gaussian shape and applies the command **bandw2** with the parameter total rotation = 90° and stores the result of the calculation into the char [ ] result. The return value of **analyzeShapeC** is the number of results calculated. In the case of **bandw2** only one result, the bandWidthFactor Δω * ΔT is returned. Now the Bandwidth factor can be used for further calculations in the AU program.

Most of the analyze commands return more than one result. These are then separated by semicolons in the returned char [ ] result. The while loop in the example scans this array and copies the single results to the double array retValue. Now all results of the analyze command are available for further calculations.

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <limits.h>
#include <ShapeIO/ShapeIOC.h>
#define MAX_ANZ 20
char result[PATH_MAX];
char filename[PATH_MAX];
char *singleResult;
int i, numbOfParams;
double retValue[MAX_ANZ];
double bandWidthFactor;


/* read gaussian shape and call analyze command bandw2 */ (void)sprintf(filename, "%slists/
wave/Gauss", getstan(0, 0));


numbOfParams = analyzeShapeC(filename, "bandw2", "90", &result[0]);
if (numbOfParams <= MAX_ANZ)
{
i = 0;
 /* scan result string for results */
singleResult = strtok(result, ",");
retValue[i] = atof(singleResult);


while ((singleResult = strtok(0, ", ")))
{
retValue[++i] = atof(singleResult);
if (i >= numbOfParams)
break;
}
```

```
bandWidthFactor = retValue[0];
}
.....
.....
deleteShapeC();
QUIT
```

7.2 The C header file ShapeIOC.h

```
/**
* reads shape in specified file into internal memory
*
* \param fileName full path of the shape file
*
* \return pointer to the amplitude array of size nPoints
* \return nPoints number of points
* \return phasePtr pointer to the phase array of size nPoints
* if (*phasePtr == 0) no phase data is present (gradient)
* \return minAmp minimum amplitude of the shape
* \return maxAmp maximum amplitude of the shape
* \return integFac integral factor of the shape
*/
extern DLL_INTERFACE const double* readShapeC (const char* fileName, int* nPoints,
const double** phasePtr, double* minAmp, double* maxAmp, double* integFac);


/**
* writes amplitude and phase data to specified file
* if a shape is present in memory its excitation mode is taken
*
* or
*
* writes shape in internal memory to specified file
*
* does not change the shape held in memory
*
* \param fileName full path of the shape file
* \param dataPtr pointer to the 2*nPoints array of amplitude and phase values
* or nPoints array of amplitude values (gradient)
* if (dataPtr==0) shape held in memory is written to disc
* \param nPoints number of points
* \param phaseData 0 - no phase data (gradient), !0 - phase data
* \param bwFac bandwidth factor of the shape
* \param shapeMode number of offset frequencies the shape is modulated
*
```

```
* \return 0 if operation failed, 1 if it succeeded
*/
extern DLL_INTERFACE int writeShapeC (const char* fileName, const double* dataPtr, int nPoints, int phaseData, double bwFac, int shapeMode);

/**
* generates the specified shape into internal memory
*
* \param shapeName name of the shape to generate
* \param params parameters of the shape
* \param phasedata 0 - no phase data (gradient), !0 - phase data
*
* \return pointer to the amplitude array of size 2*nPoints or
* nPoints (gradient)
*/
extern DLL_INTERFACE const double* generateShapeC (const char* shapeName, const char* params, int phaseData);

/**
* analyses the shape in specified file
*
* or
*
* analyses the shape held in memory
*
* does not change the shape held in memory
*
* \param fileName name of the shape file
* if (fileName==0) shape held in memory is analyzed
* \param command analyze operation
* \param params parameters for the operation
*
* \return number of result parameters
* \return result results of the calculations
*/
extern DLL_INTERFACE int analyzeShapeC (const char* fileName, const char* command, const char* params, char* result);

/**
* reads and manipulates the shape in specified file
*
* or
*
* manipulates the shape held in memory
```

```
*
* \param fileName name of the shape file
* if (fileName==0) shape held in memory is manipulated
* \param command manipulation operation
* \param params parameters for the operation
* \param freqList parameters for the operation
*
* \return pointer to the amplitude array of size nPoints
* \return phasePtr pointer to the phase array of size nPoints
* if (*phasePtr == 0) no phase data is present (gradient)
* \return nPoints number of points
* \return shapeMode number of offset frequencies the shape is modulated with
*/
extern DLL_INTERFACE const double* manipulateShapeC (const char* fileName, const char* command, const char* params, const char* freqList, const double** phasePtr, int* nPoints, int* shapeMode);


/**
* adds 2 shapes and writes the result to disc
* leaving shape held in memory unchanged
*
* or
*
* adds 2 shapes and stores result into memory
*
* \param inputFile1 name of the 1st shape file
* \param inputFile2 name of the 2nd shape file
* \param outputFile name of the output file
* if (outputFile==0) result is put into memory
*
* \return 0 in case of success
*/
extern DLL_INTERFACE int addShapesC (char* inputFile1, char* inputFile2, char* outputFile);


/**
* adds several shapes and writes the result to disc
* leaving shape held in memory unchanged
*
* or
*
* adds several shapes and stores result into memory
*
* \param inputFiles space or comma separated file names
```

```
* \param outputFile name of the output file
* if (outputFile==0) result is put into memory
* \param scaleF array of size numberOfFiles with scale factors
* \param alignment 0, 1, 2 - left, center, right alignment

*
* \return 0 in case of success
*/
extern DLL_INTERFACE int addShapesMultC (const char* inputFiles, const char* outputFile,
double* scaleF, int alignment);

/**
* merges 2 shapes and writes the result to disc
* leaving shape held in memory unchanged
*
* or
*
* merges 2 shapes and stores result into memory
*
* \param inputFile1 name of the 1st shape file
* \param inputFile2 name of the 2nd shape file
* \param outputFile name of the output file
* if (outputFile==0) result is put into memory
*
* \return 0 in case of success
*/
extern DLL_INTERFACE int mergeShapesC (char* inputFile1, char* inputFile2, char*
outputFile);

/**
* merges several shapes and writes the result to disc
* leaving shape held in memory unchanged
*
* or
*
* merges several shapes and stores result into memory
*
* \param inputFiles space or comma separated file names
* \param outputFile name of the output file
* if (outputFile==0) result is put into memory
*
* \return 0 in case of success
*/
extern DLL_INTERFACE int mergeShapesMultC (const char* inputFiles, char* outputFile);
```

```
/**
* returns the integral factor of the shape in specified file
*
* or
*
* returns the integral factor of the shape held in memory
*
* does not change the shape held in memory
*
* \param fileName name of the shape file
* if (fileName==0) shape held in memory is analyzed
*
* \return integral factor of the shape
*/
extern DLL_INTERFACE double getIntegFacC (const char* fileName);


/**
* clears the shape from internal memory if present
*/
extern DLL_INTERFACE void deleteShapeC ();


/**
* !!! Depricated! Use readShapeC instead !!!
*
* reads ASCII formated shape file
* removes any shapes from memory and keeps the read shape in memory
*
* \param fileName full path of the shape file
* \param exMode excitation mode of the shape
* \param shapeMode number of offset frequencies the shape is modulated with
*
* \return pointer to the amplitude array of size 2*nPoints or
* nPoints (gradient)
* \return nPoints number of points
* \return phasedata 0 - no phase data (gradient), !0 - phase data
*/
extern DLL_INTERFACE const double* readShapeASCIIC (const char* fileName, int*
nPoints, int* phaseData, int exMode, int mode);
```

8 Shape File Formats

8.1 Shape File Format ASCII

The ShapeTool is able to cope with plain ascii files to import user manufactured shapes. To indicate whether the data has phase information the keywords “RFVERSION_F” and “GRADIENT” are used respectively.

No meta information is contained in the file which restricts the use of features within the ShapeTool. An example of how a shape file looks like is shown below:

RF-shapes	Gradients
RFVERSION_F	GRADIENT
-5.14574 0.0000	1.000000e-02
…	…
-5.16086 0.0000	1.000000e-02

8.2 Shape File Format Version 1.0

The standard shape file format for the ShapeTool before Topspin version 3.5. An example is shown below:

##TITLE= /tshome/exp/stan/nmr/lists/wave/Gauss
##JCAMP-DX= 5.00 Bruker JCAMP library
##DATA TYPE= Shape Data
##ORIGIN= Bruker Analytik GmbH
##OWNER= <>
##DATE= 00/03/23
##TIME= 08:19:08
##$SHAPE_PARAMETERS= Type: Gauss ; Truncation Level: 1
##MINX= 1.000000e+00
##MAXX= 9.999954e+01
##MINY= 0.000000e+00
##MAXY= 0.000000e+00
##$SHAPE_EXMODE= Universal
##$SHAPE_TYPE=
##$SHAPE_USER_DEF=
##$SHAPE_REPHFAC=
##$SHAPE_TOTROT= 9.000000e+01
##$SHAPE_BWFAC= 2.122000e+00
##$BWFAC50 =
##$SHAPE_INTEGFAC= 4.115776e-01
##$SHAPE_MODE= 0
##NPOINTS= 1000
##XYPOINTS= (XY..XY)

1.000000e+00, 0.000000e+00

1.018591e+00, 0.000000e+00

1.200269e+00, 0.000000e+00

...

1.037490e+00, 0.000000e+00

1.018591e+00, 0.000000e+00

1.000000e+00, 0.000000e+00

##END=

8.3 Shape File Format Version 2.0

The standard shape file format for the ShapeTool since Topspin version 3.5. The shape is fully parameterized which allows full control of composition and subsequent processing of up to 16 sub shapes contained in one shape. An example is shown below:

##TITLE= Parameter file, TopSpin 3.5

##JCAMPDX= 5.0

##DATATYPE= Parameter Values

##ORIGIN= Bruker BioSpin GmbH

##OWNER=

##MINX= 1.00000500

##MAXX= 100.00000000

##MINY= 0.00000000

##MAXY= 0.00000000

##$SHAPE_ADDPHASE= 0

##$SHAPE_AMP= 100

##$SHAPE_BWFAC= 2.122

##$SHAPE_BWFAC50= 0

##$SHAPE_CALCBW= yes

##$SHAPE_CALCINTEG= yes

##$SHAPE_EXMODE= <Universal>

##$SHAPE_EXPAND= <>

##$SHAPE_EXPANDCF= (0..143)

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

##$SHAPE_EXPFAC= 0

##$SHAPE_FRACEND= 0

##$SHAPE_FRACORG= 0

##$SHAPE_FRACSTART= 0

##$SHAPE_INTEGFAC= 0.411579478606509

##$SHAPE_KEEPRES= yes

##$SHAPE_LENGTH= 1500

##$SHAPE_MAN= <>
##$SHAPE_MODE= 0
##$SHAPE_MODFREQ= 0
##$SHAPE_MODPHASE= 0
##$SHAPE_MODSCALE= 0
##$SHAPE_MODSW= 0
##$SHAPE_MODSWDIR= -1
##$SHAPE_NPOINTS= 1000
##$SHAPE_OFFS= <>
##$SHAPE_OFFSCOEFF= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHAPE_PARAMETER= <>
##$SHAPE_REPHFAC= 0
##$SHAPE_RES= 0
##$SHAPE_SCALEFAC= 1
##$SHAPE_SCALETYPE= 0
##$SHAPE_SPOAL= 0
##$SHAPE_SPOFFS= 0
##$SHAPE_SPOPH= 0
##$SHAPE_TOTROT= 90
##$SHAPE_TYPE= <Excitation>
##$SHAPE_USER_DEF= 0
##$SHAPE_VERSION= 2.1
##$SHL_ADDPHASE= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_ALIGNOFFSET= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_AMP= (0..15)
100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_AMPFAC= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_BETA= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_BWFAC= (0..15)
2.122 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_CALCBW= (0..15)
yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
##$SHL_CALCINTEG= (0..15)
yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
##$SHL_CENTERAMP= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_CYCNUM= (0..15)

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_DIPOLCPL= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_ENDAMP= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXMODE= (0..15)
<Universal> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_EXPAND= (0..15)
<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_EXPANDCF0= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF1= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF10= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF11= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF12= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF13= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF14= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF15= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF2= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF3= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF4= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF5= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF6= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF7= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF8= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPANDCF9= (0..143)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_EXPFAC= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_FRACEND= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_FRACORG= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_FRACSTART= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_FULLHALF= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_FUNCAMP= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_FUNCTION= (0..15)
<Gauss> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_GCO= (0..15)
<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_GCOCOEFFS0= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS1= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS10= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS11= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS12= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS13= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS14= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS15= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

##$SHL_GCOCOEFFS2= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS3= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS4= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS5= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS6= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS7= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS8= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_GCOCOEFFS9= (0..59)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_INTEGFAC= (0..15)
0.411579478606509 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_LB= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_LEFTLIMIT= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_LENGTH= (0..15)
1500 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MAN= (0..15)
<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_MEANAMP= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MODAMP= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MODE= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MODFREQ= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MODPHASE= (0..15)

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MODSCALE= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MODSW= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_MODSWDIR= (0..15)
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
##$SHL_MU= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_NPOINTS= (0..15)
1000 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFS= (0..15)
<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_OFFSCOEFF0= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF1= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF10= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF11= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF12= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF13= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF14= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF15= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF2= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF3= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0

##$SHL_OFFSCOEFF4= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF5= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF6= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF7= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF8= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_OFFSCOEFF9= (0..47)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_PHASE= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_PHASEFAC= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_POWERINDEX= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_QUADCOEFF= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_RIGHTLIMIT= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SCALEFACTOR= (0..15)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##$SHL_SF= (0..15)
<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_SFCOEFFS0= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS1= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS10= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS11= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS12= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS13= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS14= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS15= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS2= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS3= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS4= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS5= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS6= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS7= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS8= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SFCOEFFS9= (0..200)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SMOOTHED= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SMOOTHING= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SPOAL= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SPOFFS= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SPOPH= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_STARTAMP= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SW= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_SWDIR= (0..15)
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
##$SHL_TANKAPPA= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_TOTROT= (0..15)
90 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_TRUNCLEV= (0..15)
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_TYPE= (0..15)
<Excitation> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
##$SHL_WINDOWSIZE= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##$SHL_ZETA= (0..15)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

```
##NPOINTS= 1000
##XYPOINTS= (XY..XY)
1.000005, 0.000000
1.018596, 0.000000
1.037495, 0.000000
...
1.037495, 0.000000
1.018596, 0.000000
1.000005, 0.000000
##END=
```

Shape File Formats

9 Contact

Manufacturer

Bruker BioSpin GmbH
Silberstreifen 4
D-76287 Rheinstetten
Germany
http://www.bruker.com

WEEE DE43181702

NMR Hotlines

Contact our NMR service centers.

Bruker BioSpin NMR provides dedicated hotlines and service centers, so that our specialists can respond as quickly as possible to all your service requests, applications questions, software or technical needs.

Please select the NMR service center or hotline you wish to contact from our list available at:

https://www.bruker.com/service/information-communication/helpdesk.html

Phone: +49 721-5161-6155

E-mail: *nmr-support@bruker.com*

Bruker Corporation

info@bruker.com
www.bruker.com

Order No: H146199