SpinCAD

UNITY INOVA NMR Spectrometer Systems

Pub. No. 01-999145-00, Rev. A0900

VARIAN

SpinCAD
UNITY*INOVA* NMR Spectrometer Systems
Pub. No. 01-999145-00, Rev. A0900

Revision history:
Rev. A0900 – VNMR 6.1C Beta release

Applicability of manual:
UNITY*INOVA* NMR spectrometer systems with VNMR 6.1C

Technical contributors: Boban John, Krish Krishnamurthy, Matt Howitt, Debra Mattiello, Evan Williams
Technical writer: Michael Carlisle
Technical editor: Dan Steele

3120 Hansen Way, Palo Alto, California 94304
http://www.varianinc.com

Table of Contents

Chapter 1. Getting Started 9

1.1 Introduction 9
1.2 Starting SpinCAD 10
1.3 Creating a New Pulse Sequence 10
1.4 Creating and Naming Multiple Channels 11
1.5 Building the S2PUL_SC Pulse Sequence 13
1.6 Creating a Phase Table 16
1.7 Saving the S2PUL_SC Pulse Sequence 19
1.8 Running the S2PUL_SC Pulse Sequence 19

Chapter 2. Reference 21

2.1 Manipulating the SpinCAD Window 21
2.2 Main Menu 21
Sequence Menu: Loading and Saving Pulse Sequences 21
Elements Menu: Showing Element Components 22
Tools Menu: Using SpinCAD Tools 23
Labels Menu: Showing Labels on the Drawing Canvas 27
Help Menu 27
2.3 Pulse Sequence Information 27
Pulse Sequence 27
Number of Channels 28
Toolbox 29
Pulse Element Attributes 30
Docking Tool 30
Navigation Arrows 31
Drawing Canvas 31
Placing Elements on the Drawing Canvas 31
Phase Table Area 32
Trash 32
Docking 32
Phases 34
2.4 Gradients Values and Calibration 39
Calibration Parameters 40
Calibration Tables 40
Horizontal Bore Imaging Systems 40
2.5 Setting Overhead Delay Time 41
2.6 The Collection Manager 41
Understanding Collections 41
Making a Pulse Sequence Collection 42
Closing the Collection Manager 42
Renaming a Collection 42
Deleting a Collection 42

Chapter 3. Expressions and Calculations 43
3.1 Programming Overview 43
3.2 Table-Driven Approach 44
Tables and Flow Control Statements 44
3.3 Variables 46
Variable Declarations 46
Using COMPVALUE for Local Variables 46
3.4 Operators and Expressions 47
Math Operations 47
3.5 Program Control Flow 48
3.6 Function Calls 48
Math Functions 48
String Functions 50
Special Functions 50
3.7 Special Operations 50
Outputting Values and Information 50
Executing Programs 51

Chapter 4. Translating C Pulse Sequences to SpinCAD Format 53
4.1 vnmr2sc Description and Features 53
Construction and Storage of A New Pulse Sequence 53
C PSG Functions and Features that Are Converted 54
Other Implemented Features 55
4.2 Limitations in vnmr2sc 55
Inherent Limitations 55
Features that Are Currently Not Implemented 56
4.3 Using vnmr2sc 57
Conditions 57
Converting a C Sequence 58
4.4 Cleaning Up Converted Pulse Sequences 59
Checking Converted Sequences 59

Appendix. List of SpinCAD Components 63
Common Attributes 63
System Collections 63
Prefix 63
Pulse Elements 64
Primitives 76

Index 87

List of Figures

Figure 1. S2PUL_SC Pulse Sequence 9
Figure 3. Pulse Sequence Name Field 10
Figure 2. SpinCAD Window 11
Figure 4. Number of Channels Menu 11
Figure 5. Channel Label 12
Figure 6. Channel Attributes Tool 12
Figure 7. Channel Attributes Toolbox 13
Figure 8. Components List 13
Figure 9. Delay 14
Figure 10. Pulse Element (Delay) Definition Tool 14
Figure 11. Pulse Docked to Delay 15
Figure 12. Docking Tool 15
Figure 13. Pulse Element (Pulse) Attributes List 16
Figure 14. Phase Table 16
Figure 15. Docked Phase Table 17
Figure 16. Pulse Element (Phase Table) Attributes List 17
Figure 17. Receiver Phase 17
Figure 18. Docked Receiver Phase 18
Figure 19. Pulse Element (Receiver Phase) Attributes List 18
Figure 20. S2PUL_SC Pulse Sequence 18
Figure 21. Assignment Tool in dg Window 19
Figure 22. Sequence Window 22
Figure 23. Elements Menu 22
Figure 24. Element Attributes 23
Figure 25. Tools Menu 23
Figure 26. PSG Audio/Visual Control Window 25
Figure 27. Print Window 25
Figure 28. PostScript File in PageView Window 26
Figure 29. Zoom Window 26
Figure 30. Labels Window 27
Figure 31. Labels 27
Figure 32. Update/Hide Labels Window 27
Figure 33. Pulse Sequence Field 28
Figure 34. Comment Field 28
Figure 35. Channel Attribute Fields 29
Figure 36. Toolbox 29
Figure 37. Attribute Definition Tool 30
Figure 38. Docking Tool 30
Figure 39. Drawing Canvas 31
Figure 40. Phase Tables Area 32

Figure 41. Actual Pulse Width 33
Figure 42. Programming in rof1 33
Figure 43. Overlapping an Event 34
Figure 44. Phase Table Menu Tab 35
Figure 45. Phase Table Menu 36
Figure 46. Collection Manager 41
Figure 47. Euler Angles 67

List of Tables

Table 1. Map of Phase(n) Values 37
Table 2. Values Providing Quadrature 38
Table 3. Dimensions and Parameters 38
Table 4. Calibration Parameters 40
Table 5. Variable Types 46
Table 6. Supported Operators 47
Table 7. Unsupported Operators 48
Table 8. Supported Math Functions 49

Chapter 1. Getting Started

Sections in this chapter:

- 1.1 “Introduction” on page 9
- 1.2 “Starting SpinCAD” on page 10
- 1.3 “Creating a New Pulse Sequence” on page 10
- 1.4 “Creating and Naming Multiple Channels” on page 11
- 1.5 “Building the S2PUL_SC Pulse Sequence” on page 13
- 1.6 “Creating a Phase Table” on page 16
- 1.7 “Saving the S2PUL_SC Pulse Sequence” on page 19
- 1.8 “Running the S2PUL_SC Pulse Sequence” on page 19

This chapter is a brief tutorial to help familiarize you with SpinCAD. In this tutorial, you will create the S2PUL_SC pulse sequence shown in Figure 1.

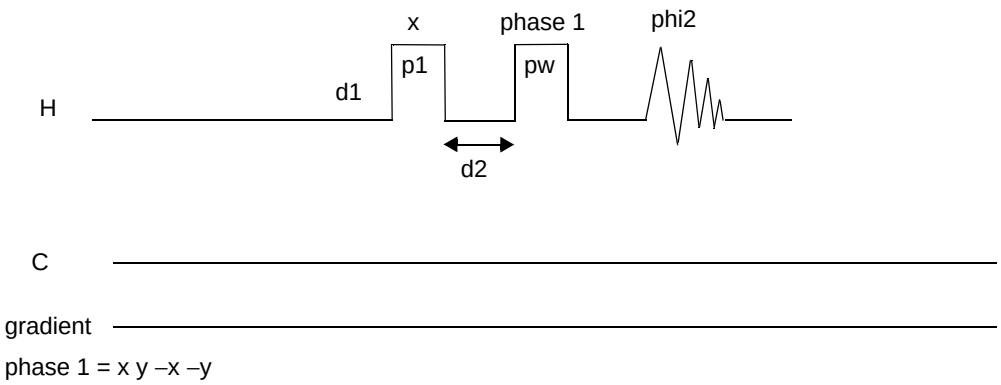


Figure 1. S2PUL_SC Pulse Sequence

1.1 Introduction

SpinCAD™ is a new graphical pulse sequence programming environment for UNITY*INOVA*. It enables you to create a pulse sequence by dragging and dropping familiar NMR elements, such as a pulse, gradient, and delay. At another level, it provides a mechanism to directly

set the spectrometer hardware, such as turning on a TTL gate. SpinCAD also enables you to build your own collections of elements for reuse in different pulse sequences. For example, you might build a WATERGATE segment, which can then be dragged out and connected to any sequence.

SpinCAD provides a convenient mechanism for specifying events on different RF channels. Events can be constructed as a serial class of events or as events that occur in parallel. SpinCAD also uses a completely new model for representing the pulse sequence. As a result, the amount of disk space needed to hold the sequence has been reduced from tens of megabytes to tens of kilobytes for 2D, 3D, and 4D experiments. The starting time for these experiments has been reduced to seconds.

SpinCAD enables you to array shapes and pulse shapes “on-the-fly” from within a pulse program, using `Pbox`.

The `dps` program can display SpinCAD sequences. A new option to `dps` enables you to view the fine details of a SpinCAD pulse sequence. In this mode, all the primitive hardware events are shown. This mode is selected with `dps('-fine')`.

Although SpinCAD provides all the capabilities of the earlier C-based pulse programming methodology, it does not preclude the use of C sequences, which can be used in conjunction with SpinCAD.

There is also a tool, `vnmrs2c`, that converts a C sequence into SpinCAD.

In general, SpinCAD uses the same parameters as do the C sequences. However, there are two significant differences:

- The incremented delays `d2`, `d3`, and `d4` are not used in SpinCAD. Instead SpinCAD uses `t1`, `t2`, etc.
- The `phase`, `phase2`, etc. parameters have slightly different values. In particular, the value 5 inverts a designated gradient pulse.

1.2 Starting SpinCAD

In the VNMR input window, enter **`seqfil='s2pul_sc'`** then the command **`spincad`**. The SpinCAD window shown in Figure 2 opens.

1.3 Creating a New Pulse Sequence

To create a new pulse sequence, do the following procedure:

1. Click on **Sequence** in the main menu, then select **New sequence**.
2. Enter **`s2pul_sc`** in the **Pulse Sequence** field, as shown in Figure 3.

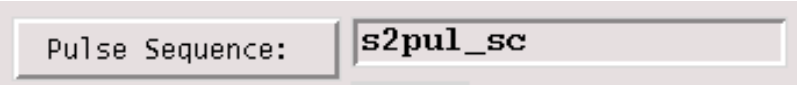


Figure 3. Pulse Sequence Name Field

After you have named the sequence, the next step is to create three channels.

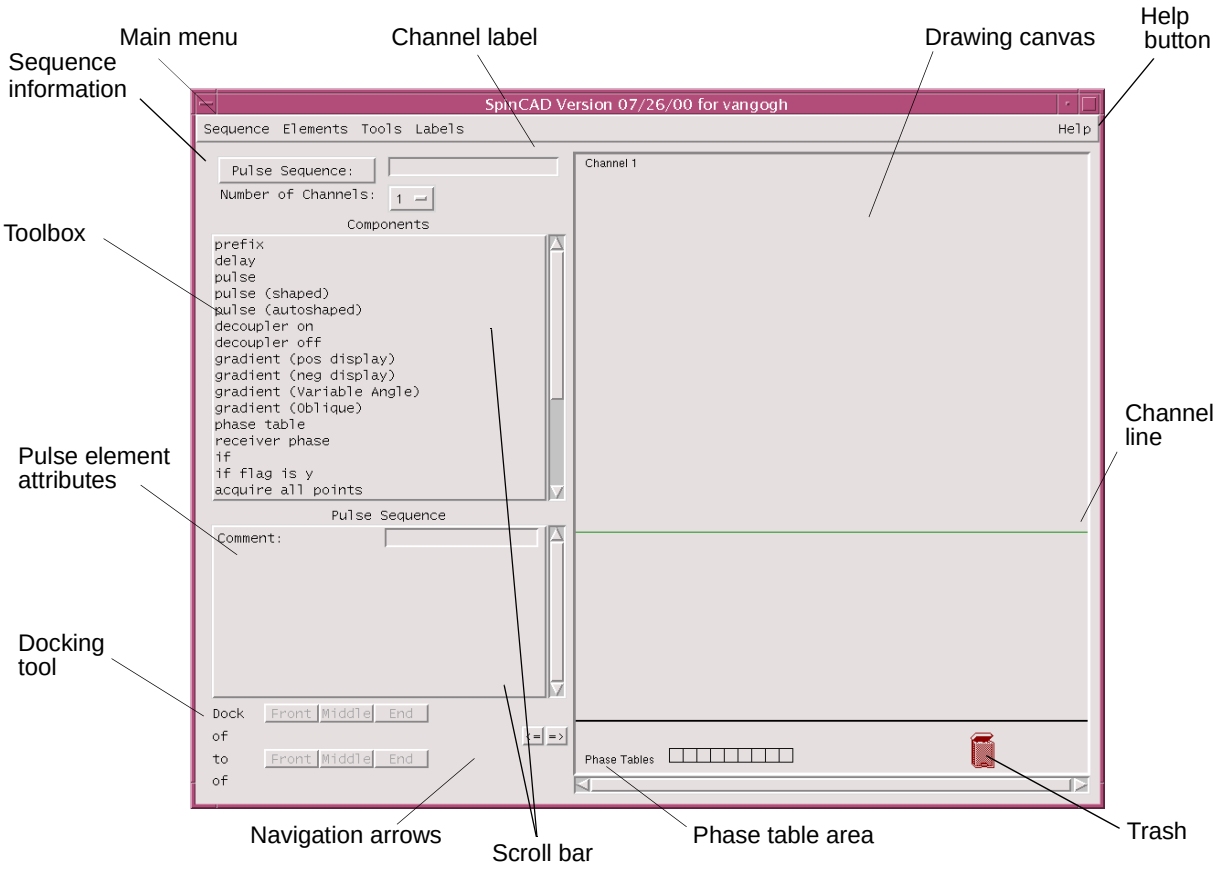


Figure 2. SpinCAD Window

1.4 Creating and Naming Multiple Channels

SpinCAD enables you to create up to eight channels on the drawing canvas. You can also name and define each channel. Do the following procedure to create three channels named Observe, Decoupler, and Gradient:

3. From the **Number of Channels** pull-down menu, shown in Figure 4, select **3**.

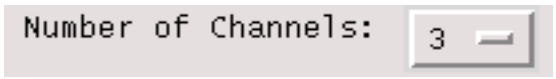


Figure 4. Number of Channels Menu

4. Rename the Channel 1 label to Observe:

a. In the drawing canvas, click on the **Channel 1** label shown in Figure 5.

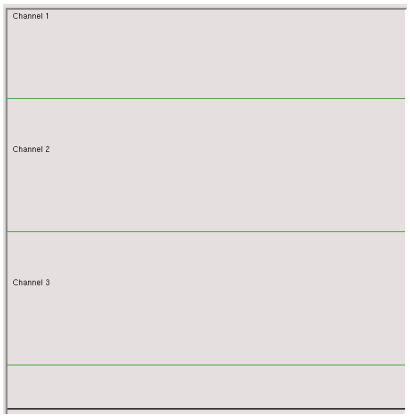


Figure 5. Channel Label

The label changes color and the Pulse Sequence toolbox changes to the Channel 1 toolbox.

b. In the Channel 1 toolbox, type **`Observe`** in the **Channel Label** field.

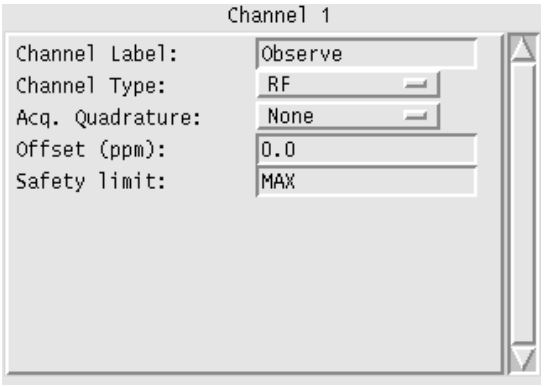


Figure 6. Channel Attributes Tool

c. Press **Enter** (or **Return**).

The Channel 1 label changes to Observe.

5. Repeat step 4 to rename the **Channel 2** label to **`Decoupler`** and the **Channel 3** label to **`Gradient`**.

In the Channel 3 attributes toolbox, shown in Figure 7, click on **RF** in the **Channel Type** field and select **Gradient** from the menu.

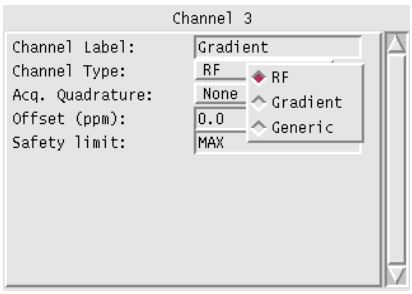


Figure 7. Channel Attributes Toolbox

1.5 Building the S2PUL_SC Pulse Sequence

To build the S2PUL_SC pulse sequence, do the following procedure:

6. From the toolbox shown in Figure 8, click on the **delay** element and drag it into the **Observe** channel.

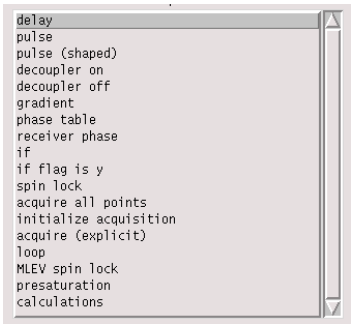


Figure 8. Components List

Place it on the channel line as shown in Figure 9.

Figure 9. Delay

7. Type **d1**. This entry appears as the Duration attribute, as shown in Figure 10:

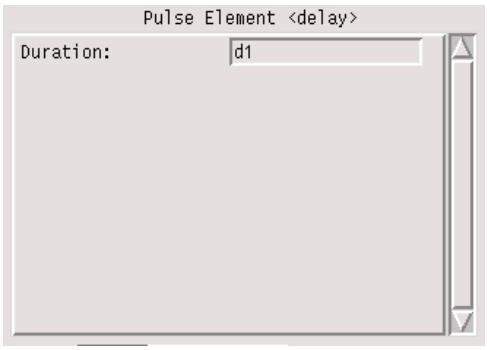


Figure 10. Pulse Element (Delay) Definition Tool

8. From the toolbox, drag a **pulse** element and dock (connect) it directly next to the delay. For more information about docking pulse elements, see section "Docking Elements" on page 32.

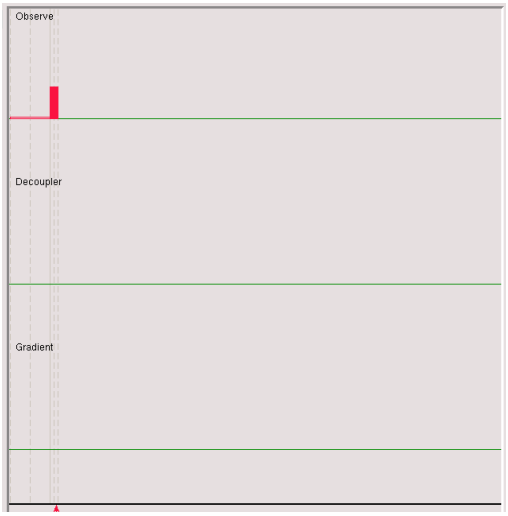


Figure 11. Pulse Docked to Delay

In the docking tool, shown in Figure 12, notice the docking setting. The **Front** of the pulse is docked to the **End** of the delay. Use the tool to change the docking order between pulse elements. To change the docking order, either click on the **Front**, **Middle**, or **End** buttons.

You can use the navigator arrows to select different sequence elements. Clicking on a sequence element with the left mouse button will also select it.

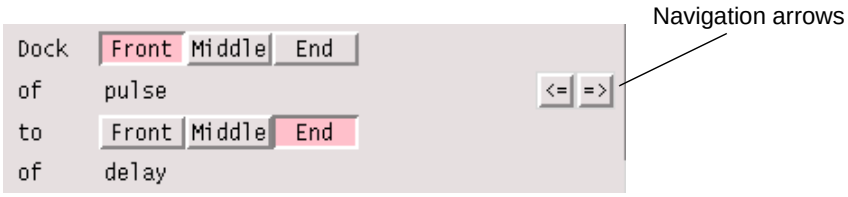


Figure 12. Docking Tool

9. In the **Pulse Element <pulse>** attributes list, enter **p1** in the **Pulse width** field and click on the arrow in the **Phase table** field and select **0**, as shown in Figure 13:

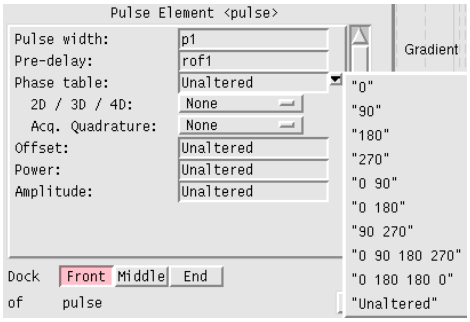


Figure 13. Pulse Element (Pulse) Attributes List

10. From the toolbox, drag a **delay** and dock the **Front** of the delay to the **End** of the pulse and enter **d2** in the **Duration** field.
11. From the toolbox, drag a **pulse** and dock the **Front** of the pulse in the **End** of the delay.
12. In the **Pulse Element <pulse>** box, enter the following attributes:

Pulse width	`pw`
Phase table	`"0 90 180 270"`

After you have built the pulse sequence, the next step is to add phase tables.

1.6 Creating a Phase Table

Some predefined phase tables are loaded when a new pulse sequence is created. This section describes how to define a new phase table.

13. From the toolbox, select **phase table**, as shown in Figure 14, drag it to the Phase Tables area.

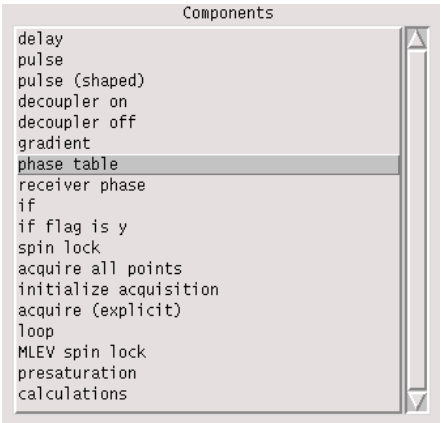


Figure 14. Phase Table

To see the attributes of a predefined phase table in the **Pulse Element** attributes box, click on one of the phase table symbols shown in Figure 15.

14. Dock the **Front** of the phase table to the **End** of the right-most phase table, as shown in Figure 15.

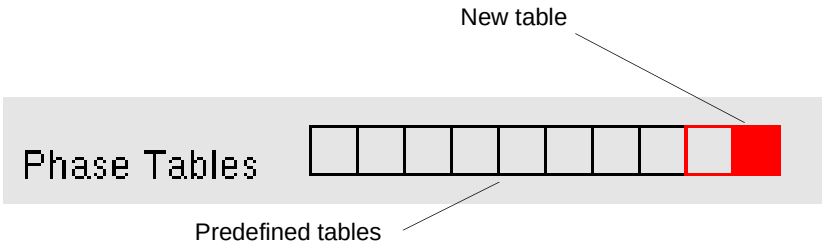


Figure 15. Docked Phase Table

15. In the **Pulse Element <phase table>** attributes list, shown in Figure 16, enter the following settings in the **Table label** and **Value (degrees)** fields:

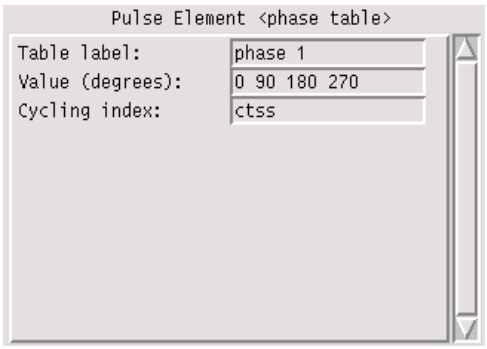


Figure 16. Pulse Element (Phase Table) Attributes List

16. From the toolbox, select **receiver phase**, as shown in Figure 17.

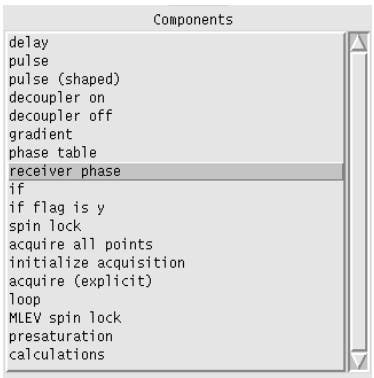


Figure 17. Receiver Phase

Drag it to the Phase Tables area and dock the **Front** of the receiver phase to the **End** of the phase table that you created in step 13 and step 15, as shown in Figure 18.

Figure 18. Docked Receiver Phase

17. In the **Pulse Element <receiver phase>** attributes list, shown in Figure 19, enter the following setting in the **Phase table** field:

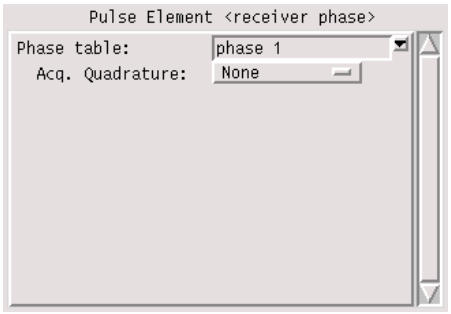


Figure 19. Pulse Element (Receiver Phase) Attributes List

18. Click on the **Labels** menu option, then **Show** to see the names of the pulse elements on the drawing canvas.

The result in the drawing canvas should resemble the pulse sequence shown in Figure 20.

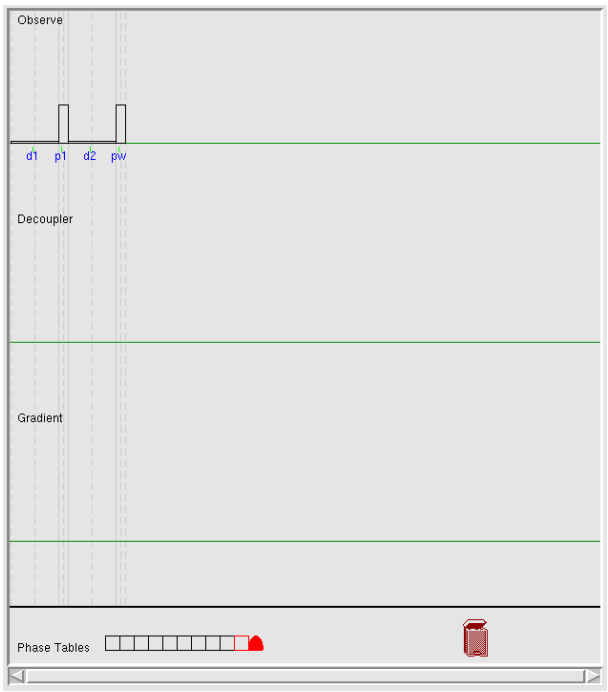


Figure 20. S2PUL_SC Pulse Sequence

1.7 Saving the S2PUL_SC Pulse Sequence

To store the S2PUL_SC pulse sequence, click on **Sequence** in the main menu, then **Save and check s2pul_sc**.

If you do not want to save your pulse sequence, click on **Exit, abandoning unsaved changes**.

If you save the pulse sequence, SpinCAD performs several checks to look for errors. The **Pulse Sequence** button is color coded during the checking process. SpinCAD first checks for attributes that are not filled in. During this check, the **Pulse Sequence** button label is yellow. If the current experiment is set up to the pulse sequence that you are editing, by having `seqfil='s2pul_sc'`, SpinCAD then performs a runtime check on the pulse sequence. During this check, the **Pulse Sequence** label is light blue. If no problems are found, the **Pulse Sequence** label returns to black. If there are problems, the label turns red and a pop-up Error Manager window appears with clickable buttons describing the problems. Clicking on a button in the window selects the problem elements.

1.8 Running the S2PUL_SC Pulse Sequence

After you have successfully built the pulse sequence, you can map the virtual channels in the sequence to sets of parameters that control the rf and gradient:

1. Click on the **Spare** tab in the VNMR **dg** window to access an assignment tool shown in Figure 21. Use this tool to map the SpinCAD rf channels to physical channels and gradients to certain directions.

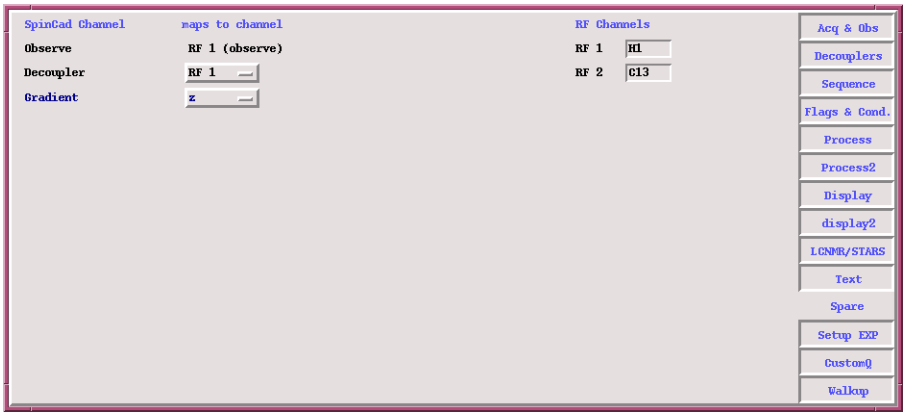


Figure 21. Assignment Tool in dg Window

2. After you have made the assignments, enter the **go** command to run the `s2pul_sc` sequence.
3. If runtime errors occur (such as illegal length delays), error messages appear in the Text tab `dg` pane.

Chapter 2. Reference

Sections in this chapter:

- 2.1 “Manipulating the SpinCAD Window” on this page
- 2.2 “Main Menu” on this page
- 2.3 “Pulse Sequence Information” on page 27
- 2.4 “Gradients Values and Calibration” on page 39
- 2.5 “Setting Overhead Delay Time” on page 41
- 2.6 “The Collection Manager” on page 41

2.1 Manipulating the SpinCAD Window

In SpinCAD, all operations are conducted within a single window that has the normal properties of a standard window, which you can reconfigure. You can enlarge and reduce its size, and the usual **Alt + . . .** operations such as “Lower,” etc. are accessible in the normal way. However, SpinCAD is not “CDE aware.” Exiting CDE while SpinCAD is running might produce a Tooltalk error and is the same as clicking **Exit, abandoning unsaved changes**. Whenever you exit SpinCAD using either **Exit, saving . . .** or the **Close** window manager function; you not only save the current work but also the geometry of the window. The only way that you can exit SpinCAD without saving its state is to click on **Exit, abandoning unsaved changes**.

SpinCAD generates several transient pop-up windows. These windows are “modeless”; that is, they do not stop activity elsewhere. In general, the windows have a **Cancel** button so that you can remove them when you no longer want them. When a window does not have a **Cancel** button, use **Close**.

2.2 Main Menu

This section describes the main menu, which consists of the submenus described in the following sections:

- “Sequence Menu: Loading and Saving Pulse Sequences” on this page
- “Elements Menu: Showing Element Components” on page 22
- “Tools Menu: Using SpinCAD Tools” on page 23
- “Labels Menu: Showing Labels on the Drawing Canvas” on page 27

Sequence Menu: Loading and Saving Pulse Sequences

In the **Sequence** menu selection, you can load:

- A new pulse sequence
- A sequence that you created and saved
- An existing sequence in SpinCAD

You can also:

- Save the current sequence that you are working on
- Save the current sequence and exit SpinCAD
- Exit SpinCAD without saving the current sequence

Tip

To permanently open the **Sequence** window, shown in Figure 22, or a window for any main menu option, click the left mouse button on the broken line above the list. Close the window by double-clicking on the button in the upper left corner of the window or by pressing the **Alt** and **F4** keys.

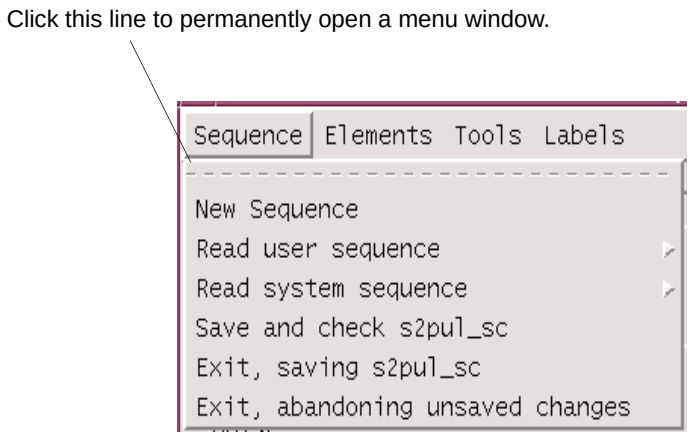


Figure 22. Sequence Window

Elements Menu: Showing Element Components

By selecting the options in the Elements menu, shown in Figure 23, you can show or hide the following items in the toolbox list:

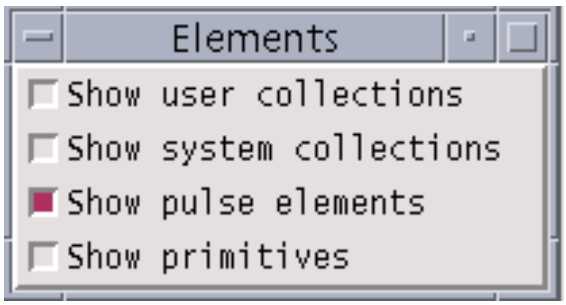


Figure 23. Elements Menu

For example, to see primitive pulse elements in the toolbox, click on **Show primitives**.

Primitives

Primitives are the basic building blocks for spectrometer control. `RF:GATE` is an example.

Composites

A composite is a single, complex element made up of one or more primitive elements. The components of a composite are represented by a single graphic object on the drawing canvas. `pulse` is an example.

Collections

A collection is a group of pulse elements saved under a single name. To see collections in the Components list, click on **Show Collections**. Section 2.6 “The Collection Manager” on page 41 contains more information about collections. `prefix` is an example.

Seeing Element Attributes

To see the attributes of an element, highlight the element by clicking on it in the drawing canvas. Its description appears in the attributes list, as shown in Figure 24.

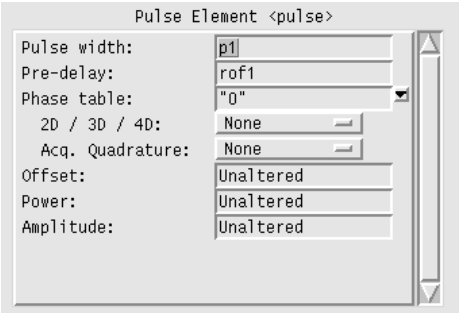


Figure 24. Element Attributes

Tools Menu: Using SpinCAD Tools

Use the **Tools** menu, shown in Figure 25, to change the appearance of the drawing canvas and perform other functions.

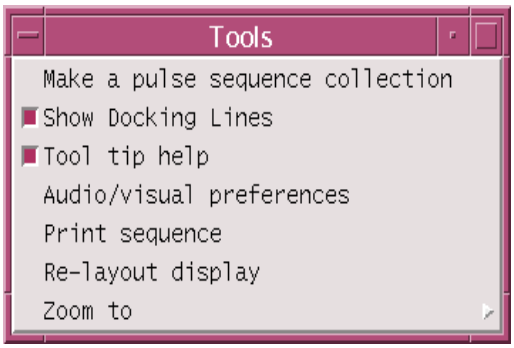


Figure 25. Tools Menu

Make a Pulse Sequence Collection

Click on this option to create a pulse sequence collection. For more information about making a collection, see the section “Making a Pulse Sequence Collection” on page 42.

Show Docking Lines

Docking lines are vertical guides for docking activities. You can use them when building a sequence. Each element has either two (Front, End) or three (Front, Middle, End) docking points. When an element is docked, points involved in docking are shown as solid, vertical lines. Unused docking points appear as dashed lines.

To see docking lines on the canvas, click on **Show Docking Lines** in the **Tools** menu. You can hide docking lines by clicking on **Show Docking Lines** a second time. While you move an element, its docking lines disappear from the drawing canvas; they reappear after it is docked. You can adjust the color of the docking lines by clicking on **Tools**, choosing **Audio/visual preferences**, and selecting **Connection lines**.

Tool Tip Help

Click on this option to see descriptions of menu items when you place the cursor on top of them. To hide tool tips, click again on this option.

Audio/Visual Preferences

You can customize the appearance of the drawing canvas by controlling the color of the following aspects of the canvas:

- Pulse elements
- A selected pulse element
- A connected pulse element
- Connection lines
- Channel lines
- Background of the drawing canvas (changing this aspect takes full effect the next time that you start SpinCAD)

Clicking on **Audio/visual preferences** opens the PSG Audio/Visual Control window, shown in Figure 26. Use the window to change the colors of all, selected, or connected pulse elements; connection and channel lines; and the background of the drawing canvas.

Controlling Sound

Use the **Audio Selection** options to control the sound volume or turn off the docking sound.

Changing Pulse Element Colors

To change the color of an element, click on a pulse element, then scroll through the list of colors in the PSG Audio/Visual Control window, shown in Figure 26, and click on a selection.

Print Sequence

When you select **Print sequence**, you can print two sets of information. Initially, all events in the sequence are numbered on the canvas. A pop-up window, shown in Figure 27, contains a text description of all the elements and their attributes, keyed to the canvas. You can print this information on your default printer.

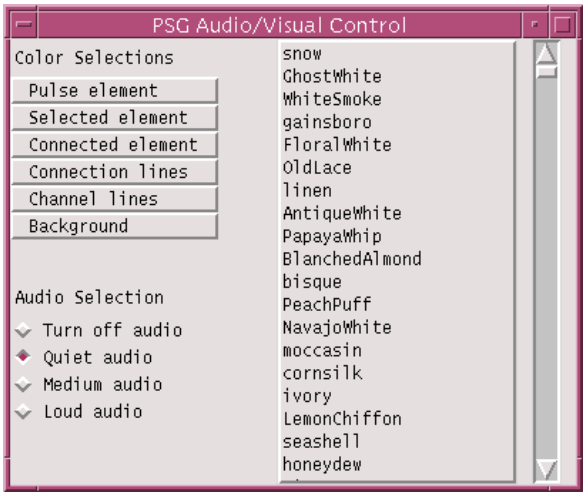


Figure 26. PSG Audio/Visual Control Window

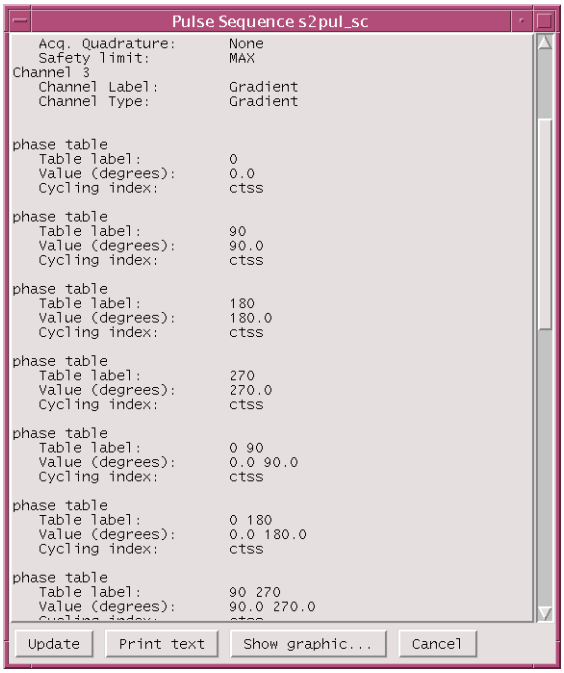


Figure 27. Print Window

You can also print the graphic canvas. If you select **Show graphic . . .,** a PageView window, shown in Figure 28, opens. In this window, a PostScript file is generated and loaded. You can inspect the result and send it to your printer. To print the result, do the following steps:

1. In the window, click the right mouse button on the **File** menu button and select **Print . . .**

   Another PageView window opens.

2. Select the desired printer and click on **Print**.

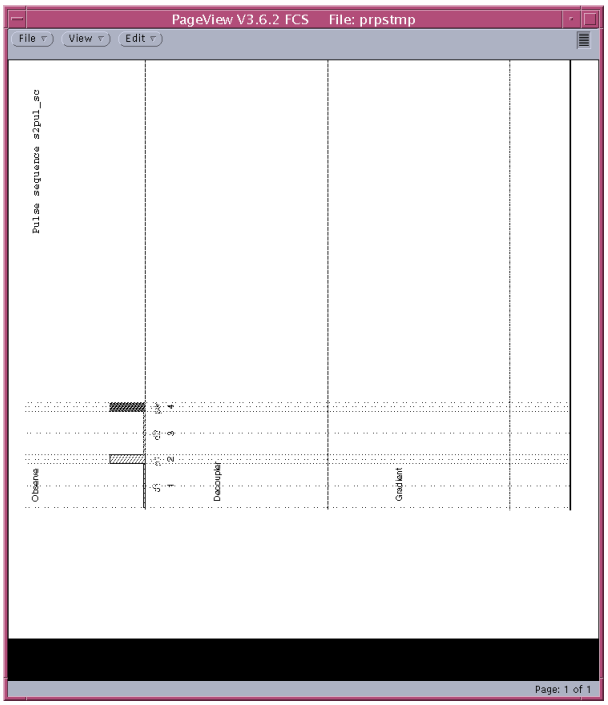


Figure 28. PostScript File in PageView Window

Re-layout Display

Clicking on **Re-layout display** enables you to change the canvas layout to optimize the use of the canvas by the sequence.

Zoom to

Clicking on the **Zoom to** option opens the window shown in Figure 29, which enables you to magnify your view of the drawing canvas time axis.

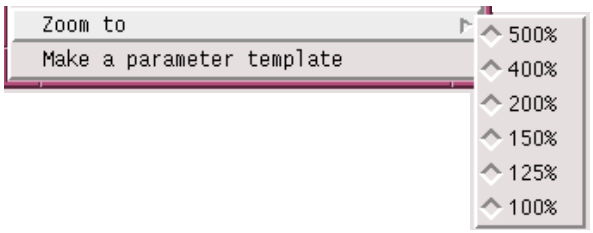


Figure 29. Zoom Window

Labels Menu: Showing Labels on the Drawing Canvas

Use the **Labels** menu, shown in Figure 30, to see labels on the drawing canvas. To see labels, select **Show**. These labels, shown in Figure 31, are color coded. Blue indicates that the label is the value of the first attribute. Green indicates the name of the object.

Figure 30. Labels Window

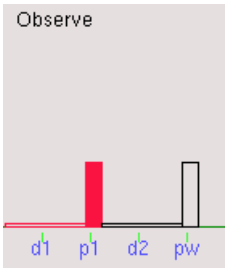


Figure 31. Labels

If you revise any labels, you can update the changes by clicking on the **Update** option, shown in Figure 32. To conceal labels from the drawing canvas, click on **Labels**, then **Hide**.

Figure 32. Update/Hide Labels Window

Help Menu

The Help menu contains getting started information describing the menu system, mouse actions, and how to edit pulse sequences. The menu also contains information about collections of pulse elements, the Collection Manager, and current collections.

2.3 Pulse Sequence Information

Pulse Sequence

The Pulse Sequence field, shown in Figure 33, is concerned with the overall pulse sequence. After you have selected and opened a pulse sequence, its name is displayed in

the entry field. When you create a new pulse sequence, use this field to enter the name of the sequence. If you exit SpinCAD and this entry field is blank, your work is saved as the sequence `noname`.

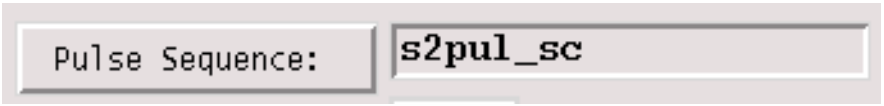


Figure 33. Pulse Sequence Field

The **Pulse Sequence** button has two functions:

- It serves as an indicator regarding the state of the current sequence.
- It gains access to the sequence comment field.

When you save a pulse sequence, the **Pulse Sequence** button is color coded during the checking process. SpinCAD first checks for attributes that are not filled in. During this check, the **Pulse Sequence** button label is yellow. If the current experiment is set up to the pulse sequence that you are editing, including having `seqfil='s2pul_sc'`, SpinCAD then performs a runtime check on the pulse sequence. During this check, the **Pulse Sequence** label is light blue. If no problems are found, the **Pulse Sequence** label returns to black. If there are problems, the label turns red. A pop-up Error Manager window might appear with clickable buttons describing the problems. Clicking on a button in the window selects the problem elements.

If you click on the **Pulse Sequence** button, the title of the Pulse Element attribute box changes to Pulse Sequence and shows the sequence **Comment** field, as shown in Figure 34, where you can enter comments. Prolific use of this feature is encouraged! You can expand this field (or any other field) by pressing the **Control** key and clicking the left mouse button in the field.

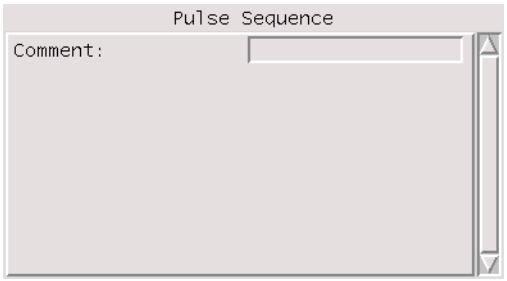


Figure 34. Comment Field

Number of Channels

Beneath the Pulse Sequence field is a button for indicating the number of channels. These channels are virtual channels and do not relate to your hardware until `go` time. SpinCAD designers have provided an arbitrary maximum of eight channels solely based on the amount of graphics real estate available. You can change the numbers of channels at any time. New channels are added at the bottom of the canvas.

Specifying Channel Attributes

Click on the **Number of Channels** button and choose a number from the list.

When you click on a channel label in the graphics canvas, the Pulse Sequence box changes to the box shown in Figure 35, so that you can specify the attributes of the channel.

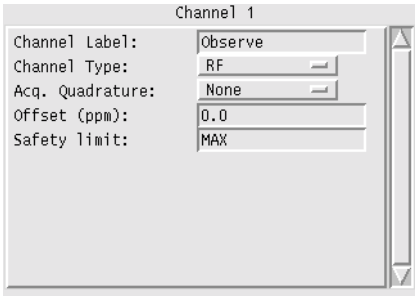


Figure 35. Channel Attribute Fields

Use the **Channel Label** field to name a channel. Click on the **Channel Type** to open a pull-down menu to specify an RF, Gradient, or Generic channel. Click on the **Acq. Quadrature** button to impose Cyclops on a pulse. You can set a **Safety limit** for any channel. The default value is `MAX`, which enables you to use the full range of power. Any other value (in units of `tpwr`) sets as global maximum for power in that channel, which is useful in decoupling.

Toolbox

Below the **Number of Channels** is the toolbox, shown in Figure 36, which is a container for currently displayed elements.

To build a sequence, click on elements and drag them out of the toolbox and onto the canvas.

The list can be longer than the displayed region; use the scroll bar to see elements that are not displayed. The order in which elements are displayed is not alphabetical. Collections are listed before composites, which are followed by primitives. Within each group, the order is arbitrary, but we have tried to put the most useful elements at the top.

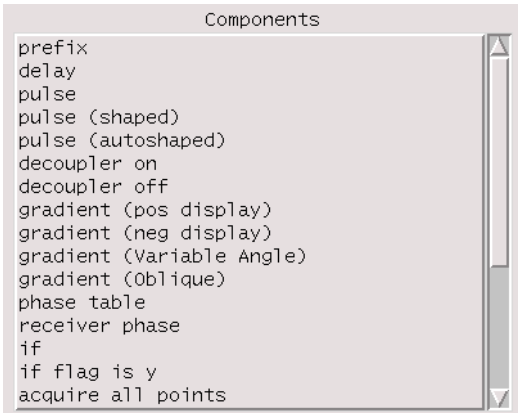


Figure 36. Toolbox

Pulse Element Attributes

Below the toolbox is a tool that enables you to see and define the attributes of a pulse element. Figure 37 shows the definition of a delay. This tool varies according to the currently selected element. If the list of attributes is too long to be displayed, use the scroll bar to see hidden attributes.

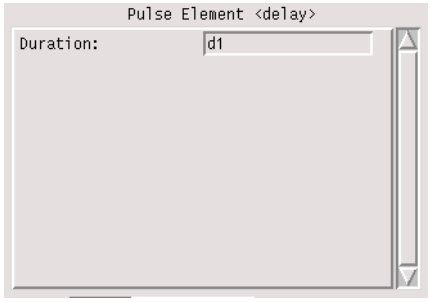


Figure 37. Attribute Definition Tool

As many attributes as possible have been preset. Those attributes with of `Unaltered` need not be filled in. However, some fields must be supplied information. These are blank and generally clustered at the top.

When you select an element, the first attribute is automatically selected. After you fill in the field, you can see the attribute on the canvas by clicking **Labels** in the main menu then **Show**. You can proceed through the attribute list by either repeatedly pressing the **Tab** key or by clicking with the mouse:

- One click places the insertion point.
- Two clicks selects a word.
- Three clicks selects a line.
- Four clicks selects the entire entry.

If you need more room to display or edit a field, press the **Control** key and click the left mouse button to open a separate edit window.

See the Appendix, “List of SpinCAD Components,” for more information about attributes.

Docking Tool

This tool, shown in Figure 38, enables you to fine-tune the docking (connecting) of one element to another element. For more information about docking, see “Docking” on page 32.

Figure 38. Docking Tool

Navigation Arrows

Each click on the navigation arrows, shown in Figure 38, enables you to move to the left or right from element to element on the drawing canvas.

Drawing Canvas

The drawing canvas, shown in Figure 39, is the center of the action in SpinCAD. This is where you compose a sequence and edit it. The key action is the manipulation of pulse elements.

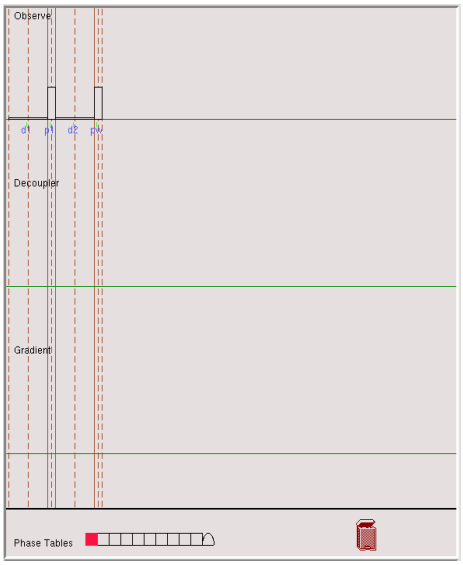


Figure 39. Drawing Canvas

Placing Elements on the Drawing Canvas

To place a pulse element on the drawing surface, do the following procedure:

1. Click on an element in the toolbox to select it then drag it onto the canvas.
2. Release the button to position the element. If the position is close enough to a docking point, the element snaps into the docking position; otherwise, select it with the mouse and drag it close to the docking point. For more information about docking elements, see the section “Docking Elements” on page 32.

Selecting Elements

When you press the right mouse button over an element, it is selected but cannot be moved.

When you select an element, it is displayed in a different color (red by default) and the attributes of that element are available for editing. The element to which it is docked is outlined in color (also red by default).

Duplicating Elements

Do the following steps to duplicate an element:

1. Select the element with the left mouse button, then press and hold the **Shift** key.

2. Drag the duplicate.

Docking Elements

When you place elements directly next to one another on the drawing canvas, you are connecting, or *docking*, the elements. A series of docked elements is called a *stack*. When you click on and drag an element in a stack, you also drag all elements that are docked to the *right* of it. You can change the docking order of stacked elements in two ways:

- In the drawing canvas, manually drag an element to the front, middle, or end of another element.
- Click on an element in the drawing canvas, then using the docking tool, shown in Figure 38, click on the **Front**, **Middle**, **End** buttons to set the docking points.

Undocking Elements

You can undock (disconnect) a selected element by selecting the left-most element in the subset of elements with the left mouse button and dragging it. When you move a docked element, all elements docked to it are also moved on the drawing canvas and the docking lines disappear. You can drag this group of elements anywhere. You can redock the group at any available dock point or remove it to the trash.

Removing Elements

To remove a subset of elements from the drawing canvas, select the left-most element in the subset of elements that you want to delete and drag the element to the trash can.

Phase Table Area

You can drag phase tables from the toolbox and add them to phase tables that are already in this area shown in Figure 40.

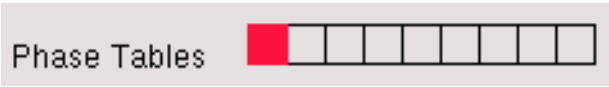


Figure 40. Phase Tables Area

Phase tables in this area are available to other elements in the pulse sequence. The main purpose of this area is to provide a visually separate area for tables.

Trash

Objects dragged into the trash cannot be recovered. After you drag an object to the trash, it is permanently gone. There is no undo option.

Docking

An element has two or three docking points. Elements are docked together by linking one element to another element at a selected pair of docking points.

Each element has a front docking point and an end docking point. Many elements also have a middle docking point. These docking points are invisible unless the element is part of the

docked pulse sequence, in which case, they are represented as vertical lines on the canvas, solid if used and dashed if not used.

Part of the runtime checking procedure is to ensure that there is a single, contiguous path from the beginning of the sequence to its end through docking points. This requirement means that there is an additional rule for docking:

If the front of an object (1) is docked to the front of another object (2), then 1 is considered to be docked into 2 and cannot extend outside 2.

Each of the two or three docking points in a composite is defined by the designer of the composite. The actual positions might not be where you first think. As an example, consider the pulse element. We think of a pulse as having a width of *pw*, but in actuality, the events that occur look more like Figure 41.

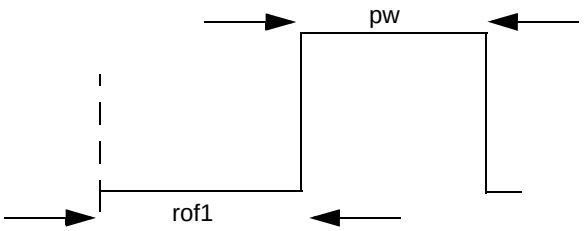


Figure 41. Actual Pulse Width

In the period *rof1*, we might unblank the amplifier, blank the receiver, set the phase, frequency, and power to be used. We really do not want to have to deal with *rof1* because it makes a simple picture complex, as shown in Figure 42B.

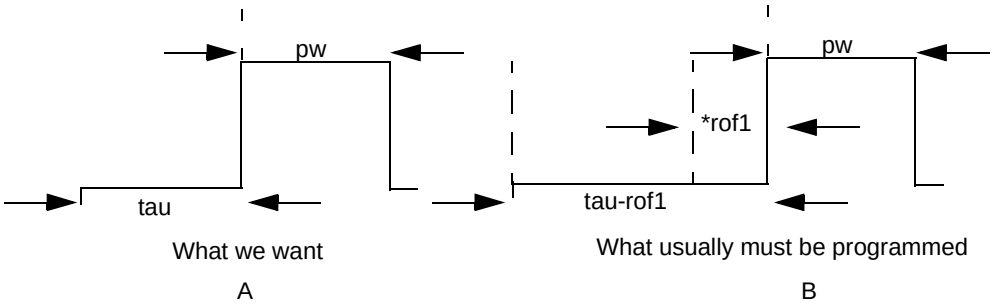


Figure 42. Programming in rof1

SpinCAD enables you to overlap events so that you can design the pulse to have a front docking point at the end of *rof1*, as shown in Figure 43.

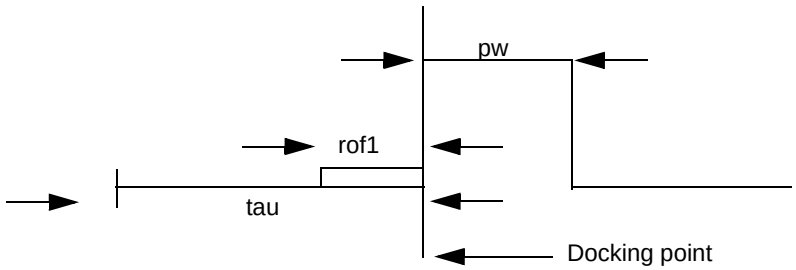


Figure 43. Overlapping an Event

The *rof1* period overlaps into the *tau* period, which raises some boundary conditions:

1. *tau*<*rof1*

   The delay is now of length *rof1*.

2. *rof1*→ 0

   The events in *rof1* overlap into *tau*.

3. *rof1*→ 0 *tau* → 0

   The minimum time is that necessary for the events in *rof1* to execute.

Phases

Phases in SpinCAD are always in degrees. Although there are both 90° and small angle phase shifting hardware, SpinCAD normally handles the two units transparently, so you don't have to.

You can specify a phase as a value, a variable, a table, or an expression. All are specified in exactly the same way because all instances end up as tables.

In general, phases are set in three places:

- As a channel attribute
- As a pulse attribute (the most common)
- As a receiver attribute

The pulse element has three phase-related attributes:

- Phase table
- 2D/3D/4D
- Acq. Quadrature

Phase Table

Phase table is an entry field with a menu tab (the down arrow shown in Figure 44.

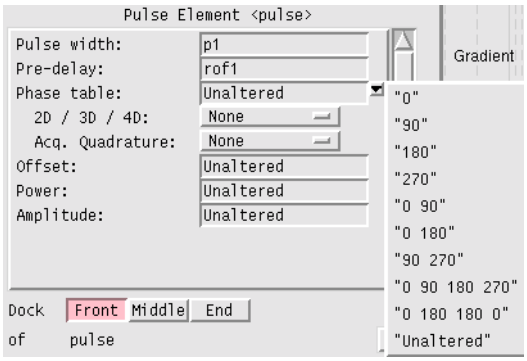


Figure 44. Phase Table Menu Tab

Clicking on the arrow opens a list of all currently known table names. When you start a new sequence (**Sequence** → **New Sequence**), a number of phase tables are automatically defined and made available. These names show the contents of each table. Note that the table names can include blanks, in which case, the use of `" "` is mandatory (e.g.,`"0 180"`). Be aware that the table name is *not* the table contents, so `"0 180"` could also be named `"spin temp.alt"`.

One special name is `Unaltered`, which is not a table but an instruction to SpinCAD to not phase shift the RF at this point. Additional special names are `fad1`, `fad2`, and `fad3`. For more information about the Phase table attribute, see the section “Elements Using Phase Tables” on this page.

2D/3D/4D

2D/3D/4D has a menu that enables you to add additional phase changes to the Phase Table attribute. The values of this attribute are `None`, `t1`, `t2`, and `t3`. These values enable you to respectively modify the phase in accordance with the value of `phase`, `phase2`, and `phase3`.

Acq. Quadrature

Acq. Quadrature also has a menu that enables you to either impose Cyclops on a pulse or to impose the channel setting. As a result, the phase applied to a pulse can be the sum of a number of logical elements, some of which can be delayed until `go` time.

Elements Using Phase Tables

The two elements that use phase tables are the pulse and the receiver phase. Selections in the Phase table menu, shown in Figure 45, have the label in double quotes, and when you click on a selection and put it into the entry field, it appears in double quotes, e.g., `"0"`, to

emphasize that the selection is a label. If the label does not include a space, +, or –, double quotes are optional.

Figure 45. Phase Table Menu

The Phase table attribute can be a label for the menu selection, it can be a parameter name, a list of one or more values, or a specification using the ^ notation. The attribute can also include the math operators + and - to combine any of the those types of labels. The following list shows some examples of legal phase table attributes (assuming the standard "prefix" list is present):

```
"0 180"
"0 180" + 90.0
fred
270.0
1^4 90^4 + "90 270" + fred
90 270
```

The parameter `fred` must exist and be set to a phase angle. If something is enclosed in `" "`, it must be a label of a table. Enclosing `fred` or 270.0 in `" "`causes an error, since no phase table with labels fred or 270.0 exists.

Note: Labels are character strings. The string `"270.0"` is not the same as `"270"`.

Several tables are predefined but not shown in the SpinCAD interface. The receiver phase element sets the oph phase table. The tables fad1 to fad5 are also made. These tables all have two elements (`0 180`) and are respectively indexed with the t1Index to t5Index variables. A pulse can have a phase table attribute set to `"0 90" + fad1`.

The use of fad1 to fad5 is checked, and any that are used are automatically added to the receiver phase oph. Therefore, fad1 to fad5 should not be explicitly added to receiver phase oph.

Tables

Tables are commonly used to describe phase cycles. Phase tables have three attributes:

- Table label
- Values (in degrees)
- Cycling index

A phase table label can be any string of characters, including spaces. The phase table values can be defined as a simple list of values or a list using the ^ notation. The list elements can be separated with either spaces or commas. Parameter names and math expression cannot be used to define the values of a phase table. The ^ notation is of the general form (`val1 val2 . . . valn`) ^ count, which causes the element within the parentheses to be duplicated `count` times. A single value does not require parentheses. The parentheses can

be nested. Only the characters `0 1 2 3 4 5 6 7 8 9 . , ( ) ^` and a space are legal. Some examples include the following strings:

Character String	*Same as*
`(0,90)^3`	`0 90 0 90 0 90`
`90^2`	`90 90`
`(0^2 (90.0 180)^2)^2`	`0 0 90.0 180 90.0 180 0 0 90.0 180 90.0 180`

The program `expandphase` expands the ^ notation. `expandphase` can be run from Unix with the table notation enclosed in " ", for example, the command

```
expandphase "((0 90)^4 (180 90)^2)^2"
```

prints two strings:

- The first string is the number of elements in the expanded list.
- The second string is the actual list of elements.

If an incorrect phase list is used as input, the first string is a negative number indicating the error and the second string is a brief description of the error, as shown in the following example.

```
slp:evan 28>expandphase "(0 90)^4 (180 90)^2"
12
0 90 0 90 0 90 0 90 180 90 180 90
slp:evan 29>expandphase "((0 90)^4 (180 90)^2)^2"
24
0 90 0 90 0 90 0 90 180 90 180 90 0 90 0 90 0 90 0 90 180 90 180 90
slp:evan 30>expandphase "(-90 90)^2"
invalid character '-'
-5
invalid element
```

`expandphase` is in `/vnmr/bin`.

Phase Control in SpinCAD

A `phase` parameter is set in VNMR to control phases. The parameter is `phase`, `phase2`, `phase3`, etc.

Table 1 shows a map of `phase(n)` values to effect on designated 2D, 3D, and 4D objects.

Table 1. Map of Phase(n) Values

phase(n) Value	*RF*	*Gradient*
`0`	+0	–
1	+0	–
2	+90	
3	tppi	–
4	+0	–
5	–	invert

Note: `phase`, not `phase1`, is for the first indirect dimension.

Table 1 shows the effects of `phase(n)` values on `t(n)`. These values are used to provide quadrature in indirect dimensions as shown in Table 2.

Table 2. Values Providing Quadrature

Phase Name	*phase(n)*
`states`	1, 2
`tppi`	3
`gradient`	4, 5 or 1,5
`firstIncr`	1
`absval`	0

A phase table element in SpinCAD selects the dimension to be used and, therefore, the set of parameters to be used. Table 3 lists the dimensions and parameter sets.

Table 3. Dimensions and Parameters

Parameter	*Value*	*Definition*
t1 Dimension		
`ni`	User defined	Number of increments
`phase`	0 to 5	Phase value for the first indirect dimension
`t1Index`	0 to (ni-1)	Increment in the first direct dimension
`t1`	1	Incrementing delay
t2 Dimension		
`ni`	User defined	Number of increments
`phase2`	0 to 5	Phase value for the first indirect dimension
`t2Index`	0 to (ni2-1)	Increment in the first direct dimension
`t2`	1	Incrementing delay
t3 Dimension		
`ni`	User defined	Number of increments
`phase3`	0 to 5	Phase value for the first indirect dimension
`t3Index`	0 to (ni3-1)	Increment in the first direct dimension
`t3`	1	Incrementing delay

A parameter can be identified in the phase object of SpinCAD to select FAD. The default value is 180. If you select `tppi`, FAD is disabled.

The following logic is used to adjust the phase.

```
IF (phase==3)
{
  fad=0.0;
  add tnIndex * 90.0
}
ELSEIF (phase==2)
```

```
{
  add 90.0
}
ELSEIF (phase==5)
{
  add 180.0
invert gradient
IF ((fad != 0.0) && ((tnIndex % 2) == 1))
  add fad
}
```

Phase Cycle Addition Order

The results of `phase(n)` and `fad(n)` are added to each phase element of the designated objects according to the increment numbers.

Cyclops is added to a phase cycle as the fastest cycling component, e.g.,

```
(0 90) + cyclops =
0 90 180 270 90 270 0 90
```

Offset

Change the offset for a channel to a new value, which produces a synthesizer frequency shift during pre-delay. The default is `Unaltered` (no change). The shape can optionally contain a linear phase ramp (SLP) to affect additional shift in frequency.

Power

As with phase, a value of `Unaltered` means that the power and amplitude or frequency does not change for this pulse.

When specified, power is in units of dB. The most commonly used parameter is `tpwr`.

Amplitude

Again, a value of `Unaltered` means that the amplitude or frequency does not change for this pulse.

Amplitude is linear modulator control. It has a range of 0-4095.

2.4 Gradients Values and Calibration

Gradient values in SpinCAD are defined in units of gauss/cm. These units are the same units that the Oblique Gradient statements use in the C type pulse sequences. In fact, the calibration parameters, tables, and procedures are the same as those used for Imaging and Triax sequences. There are three new features that make it easier to use the SpinCAD version:

- If there are no specified calibration parameters, SpinCAD assumes the gradient values are in dac units and scales the values according to the maximum dac unit for the configured gradients.
- The `gcoil` parameter, which holds the name of the gradient calibration table, can be a global parameter.

- Default PFG calibration tables are provided.

Because the gradients are specified in gauss/cm, calibration values must be defined. Use gradient tables and the `sysgcoil` and `gcoil` parameters to define calibration values. `sysgcoil` is set during configuration time to the name of a gradient calibration table residing in `/vnmr/imaging/gradtables` and should be changed whenever the gradient coil is changed. `syscoil` is defined as the current gradient coil in the system or "system gradient coil." When an experiment is joined, if that experiment has a `gcoil` parameter or if a global `gcoil` parameter exists, that `gcoil` name is set to the `sysgcoil` name and the calibration parameters are set to the values in the gradient table.

Calibration Parameters

Table 4 lists the calibration parameters.

Table 4. Calibration Parameters

`gmax`	Used when all the gradient axes are calibrated and set to the same value, which is typical on Horizontal Bore Imaging systems.
`gxmax`	Calibration parameter for the X axis.
`gymax`	Calibration parameter for the Y axis.
`gzmax`	Calibration parameter for the Z axis.

When `gcoil` is set (which happens when an experiment is joined), these values are set to the values in the gradient table that `gcoil` is set.

You can update the `gcoil` parameter (and create it if it does not exist) by executing the `updtgcoil` macro from the VNMR command line.

WARNING: **If processing is done after joining an experiment, the gradient calibration parameters are reset to the processed parameter values.**

Calibration Tables

You can create calibration tables by executing the `creategtable` macro. For more information about `creategtable`, see the *VNMR Command and Parameter Reference* manual. The macro asks for the maximum gradient strength for each of the axis, the gradient rise time, and the boresize. Tables with nominal default values are included for each of the PFG gradient sets.

When a different gradient coil is installed in the system, change the `sysgcoil` parameter to the table name defining the new coil. As `vnmr1`, change the parameter with the `config` program or use the `setgcoil` macro. If you want to do a temporary change, directly set `gcoil`. When you set `gcoil` to a new name, it runs a macro and the calibration parameters (`gmax`, `gxmax`, `gymax`, `gzmax`) are updated with their new values.

To create a global `gcoil` parameter, run `updtgcoil('global')`. Remember that with a global `gcoil` parameter, calibration parameters are created or updated for any parameter set whether or not the parameter uses gradients.

Horizontal Bore Imaging Systems

For Horizontal Bore Imaging systems, gradient calibration is usually done at installation time.

2.5 Setting Overhead Delay Time

`d0` is a parameter that defines the overhead delay time between scans and is fully implemented for SpinCAD. If `d0` does not exist, nothing is done. If `d0` does exist and is set to `'n'` (`d0='n'`), the overhead time is calculated and `d0` is set to that time. If `d0` exists and is set to a value, that value is used as the overhead time between scans. If the time is too short, an error is generated and the pulse sequence is aborted.

For more information about `d0`, see the *VNMR Command and Parameter Reference* manual or the section "Intertransient and Interincrement Delays" in Chapter 2 of the *VNMR User Programming* Manual.

2.6 The Collection Manager

You can make pulse sequence collections with the Collection Manager. The Collection Manager also enables you to add elements to a collection and rename and delete a collection. You can see instructions for using the Collection Manager by clicking on the **Show Instructions** button in the window shown in Figure 46. Hide the instructions by clicking the same button.

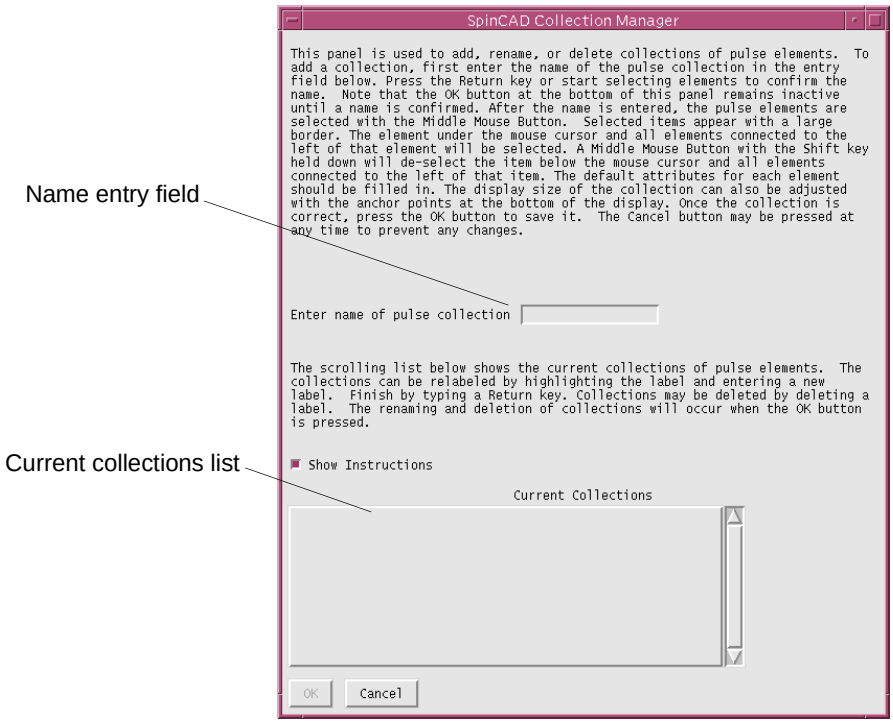


Figure 46. Collection Manager

Understanding Collections

A collection is a group of elements that are saved under a single name. You can drag a collection from the toolbox and dock it in the same way that you would drag and dock a single element. However, when you place a collection on the drawing canvas, its individual elements are shown and are individually accessible.

A collection can span more than one channel and can be docked on a channel other than its original channel. When a multichannel collection is used in a sequence, the elements distribute to the same relative channels, until the collection comes to the last channel, whereon all remaining elements are placed. For example, if a collection originally defined as having elements on channels 1, 2, and 3 is docked on channel 2, it distributes the elements onto channels 2, 3, and 4. If channel 4 does not exist, the elements destined for it appear on channel 3. After you place a collection on the drawing canvas, you can edit and rearrange it.

Making a Pulse Sequence Collection

To make a pulse sequence collection, do the following procedure:

1. Click on the **Make a pulse sequence collection** option in the **Tools** menu to open the SpinCAD Collection Manager shown in Figure 46.
2. Type the name of the pulse collection in the entry field, then press the **Return** key or start selecting elements to confirm the name.

   The **OK** button at the bottom of the panel remains inactive until you press the **Return** key.
3. Select pulse elements on the drawing canvas by clicking on them with the middle mouse button.

   The element that you click on and all elements docked to the right of that element are selected. Selected elements appear with a thick border.

   Pressing the **Shift** key and clicking the middle mouse button deselects an element and all elements docked to the right of it.

   The default attributes for each element are filled in.

   You can also adjust the display size of the collection with the anchor points at the bottom of the display.
4. When the collection is satisfactory, save it by clicking on **OK**.

Closing the Collection Manager

You can cancel any changes to a collection and close the Collection Manager by pressing the **Cancel** button.

Renaming a Collection

To rename a collection, double-click on its name in the Current Collections list, type a new name, press **Return**, then click on **OK**.

Deleting a Collection

To delete a collection, double-click on its name in the Current Collections list, press the **Delete** key, then click on **OK**.

Chapter 3. Expressions and Calculations

In general, SpinCAD enables you to avoid complex expressions written in a text-based programming language. An attribute can take a single value (e.g., `0.01`), a simple expression (e.g., `2*pw90`), or a table expression (e.g., `ph3+fad1`). However, there are occasions when a more complex expression is required. This chapter concerns the solution to this need.

Sections in this chapter:

- 3.1 “Programming Overview” this page
- 3.2 “Table-Driven Approach” on page 44
- 3.3 “Variables” this page
- 3.4 “Operators and Expressions” on page 47
- 3.5 “Program Control Flow” on page 48
- 3.6 “Function Calls” on page 48
- 3.7 “Special Operations” on page 50

3.1 Programming Overview

For the most part, expressions are programmed the same way that they are in C or Java.

When you program expressions, note these major differences from C or Java:

- Not all the operators are supported, particularly the casting, incrementing (+ +) and decrementing (`- -`) operators.
- Support for boolean variables and the operations on string variables are the Java implementation and not the C implementation.
- Arrayed variables do not need to be explicitly indexed; knowledge of their indices is embedded within the variable.

For UNITY*INOVA* and *MERCURY* systems, the acquisition subsystem is table driven. All powers, phases, and frequencies are known before the pulse sequence executes, which gives you a greater and faster measure of safety and pulse sequence display. Real-time variables are still used to perform sequence control functions and indexing into the tables.

Note: Flow control statements in this chapter do not refer to SpinCAD graphic elements but rather to C-like expression constructs.

Because the instructions are table driven, the following events occur.

- Variables declared as integers and Booleans are generally made into real-time variables unless the integers are assigned to doubles or they are involved in an expression involving strings or doubles.
- The last value assigned to a variable is the value that is used when a variable is an argument to a pulse sequence element or elements.
- Assignments within an if-else statement expand or generate a table of true and false values.

- The values of the counter in “for” loops are precalculated and used for calculations within the loop.
- “while” loops are considered to be of unknown duration and number. An expression is evaluated to a real-time expression; use it when awaiting real-time input from an outside source.

3.2 Table-Driven Approach

In general, you do not, and should not, have to worry about what is happening in the compilation of your pulse sequences into acquisition instruction codes. But having a general idea of what is happening is useful for understanding some of the limitations and more subtle aspects of the programming.

For UNITY*INOVA* systems, SpinCAD generates instructions that are “table driven,” which essentially means that all the values used to set delays, phases, powers, frequencies, and other events come from tables. There are two reasons for table-driven instructions instead of calculating values and events on the fly.

- Speed. Some calculations take a significant amount of time. Precalculating these values saves time for execution when speed might be critical.
- Safety. Knowing the values ahead of time allows SpinCAD to do some safety checking, which might range from basic boundary checks to more sophisticated power deposition checks. Precalculating also gives you a complete display of the pulse sequence.

The table-driven approach of calculations has some implication if you want to insert calculations into the pulse sequence. As has been previously stated, the syntax for calculations is similar to C or Java. The operations themselves are determined by SpinCAD. What looks like an assignment might actually be making a table of values. One rule in SpinCAD is that after a variable is used on the right side of an assignment, it should not subsequently be reset by using it on the left side of an assignment statement. Other SpinCAD rules for calculations are described in the following section.

Real-time events and variables are still used. Real-time variables are mostly used for controlling the flow of execution and indexing into tables, but can also be used for accessing external inputs. Real-time variables are not directly specified but are created if a variable is defined as an “int” and it is only used in expressions involving other real-time variables. There is no restriction on the number or names of real-time variables.

When a `go` (`ga`, `au`, `su`) program is executed, the instructions and values for the pulse sequence are calculated and sent down to the acquisition system to be executed. If the calculated values for an argument in a SpinCAD event are more than 1, they are stored in a table; otherwise, the value is stored in a general location for single values. Real-time variables are also created and stored in the general location, which is also known as the RtVar table.

When a value or values are interactively changed, these values plus all the values that depend on them are changed. Tables and/or RtVar locations are updated with the new values. The instructions are not changed, which offers a very flexible and powerful interactive capability.

Tables and Flow Control Statements

For the most part, you do not need to be concerned about tables and flow control statements. In fact, since variables have their indices incorporated into them, you do not need be

concerned about explicitly indexing into variables, exceptions are the “Flow Control” statements “for,” “if,” and "while."

for Statement

In the case of “for” statements, SpinCAD makes a table out of the *for* indices, then proceeds to step through the statements within the loop and evaluate the expressions. SpinCAD makes one pass through the loop, which has the following consequences.

In the following statements, `newmypulse` has the values: 0.0001, 0.0002, 0.0003, 0.0004, 0.0005:

```
    /* mypulse is a pulse parameter with the value of 100 us */
int nloops = 5;
double stepsize = 0.0001;
double newmypulse;
for (i=0; i<nloops; i=i+1) {
     newmypulse = mypulse+stepsize*i;
}
```

However the following statements give `newpulse` the value of 0.0001:

```
int nloops = 5;
double newmypulse = 0.0;
double stepsize = 0.0001;
for (i=0; i<nloops; i=i+1) {
     newmypulse = newmypulse+stepsize;
}
```

if Statement

To support real-time updating, the “if” statement creates values or instructions for both the true and false conditions of the if statement. So if an assignment statement is within an *if* such as in the following example:

```
   /* mypulse is a pulse parameter with the value of 100 us */
double newmypulse = 0.0;
double stepsize = 0.0001;
if (mypulse > 0.0) newmypulse = mypulse + stepsize;
```

the values for `newmypulse` are 0.0, 0.0002. The first value is if the condition is false, the second value is for if the condition is true. Since `mypulse` is greater than 0.0, the value 0.0002 is used. A consequence of this implementation is that a parameter cannot be set in more than one if statement. For example, in the following statement, `pw` is set by two separate if statements:

```
pw=0.0;
If (onflag == "y") pw=1e-6;
If (invert == "y") pw=pw*2;
```

This causes unpredictable errors. One solution is to use temporary variables. For example, the following statements work as expected:

```
pwtmp=0.0;
If (onflag == "y") pwtmp=1e-6;
pw=pwtmp;
If (invert == "y") pw=pw*2.0;
```

Another solution is to rewrite the conditional statement. For example, the following also works as expected:

```
pw=0.0;
if (onflag == "y")
  if (invert == "y")
    pw=2.0e-6;
  else
    pw=1.0e-6;
```

while Statement

The “while” statement is considered to be an indeterminate event statement: that is, it is not known up front how many times an event loops. Do not use a *while* statement in calculations; use it to wait for an event to happen.

3.3 Variables

The scope of all variables in a sequence is global within that sequence.

All VNMR parameters are considered variables and do not need to be declared. SpinCAD uses the COMPVALUE construct to enable users to use a “local” variable. Characters and the `char` type are not supported. The function `getChar` returns a character as a string from a string of characters.

Variable Declarations

When performing calculations in the `Calc` field, variables that are not in the parameter set must be declared. Table 5 lists variable types that are valid.

Table 5. Variable Types

Type	*Notes*
boolean	Boolean variables can have the values `true` or `false` and can store the results of logical expressions.
double	Double variables are generally used as arguments to pulse sequence elements.
int	Integer variables are generally converted to real-time variables and should be used as sequence control variables.
String	

These declarations can also be initialized to a value or an array of values. They can *not* be initialized to variables or expressions. These initialized values are overridden by values in the parameter set, for example:

```
double d4,d5,d6;
double p3 = 0.00001;
String sequencename = "s2pul";
double Cyclops[] = { 0.0, 90.0, 180.0, 270.0 };
```

Using COMPVALUE for Local Variables

Some SpinCAD pulse elements perform calculations in order to set various attributes of the element. Variables that are local to the element are sometimes used to hold the results of these calculations. For example, a delay might be computed as the sum of an input value for the delay plus an overhead delay for setting a piece of hardware. Local variables are

named `COMPVALUE_n_x` in which `n` is a unique number and `x` is an optional label. When viewing the pulse, you can show these variables sequence with `dps('-file')`.

3.4 Operators and Expressions

Math Operations

SpinCAD uses C operators. Table 6 lists the supported operators.

Table 6. Supported Operators

Operators	*Symbols*
assignment	=
arithmetic	+, –, *, /, %, <, <= ,>, >=, = =, !=
bitwise	~, &, |, ^
Boolean	!, &&, | |
string	+ ,= =, !=

Order of Precedence

The order of precedence for operators is the same as for C and Java and is from the highest to lowest, e.g.:

Highest
~,!,(+,– binary operators)
*,/,%
+,–
<,(=,>,>=
= =, !=
&
^
|
&&
| |
=

Precedence specifies the order in which operations are performed. Higher orders of operations are performed before lower orders of operations. If the order of precedence is the same, then the operations are performed left to right except for the binary and assignment operators.You can override operator precedence by using parentheses to group operators together. For example, when performing the following calculation:

```
e=a+b*c+d;
```

The operations are evaluated in the following order:

```
e=(a+(b*c))+d;
```

To do the additions first, group the operators like this:

```
e=(a+b)*(c+d);
```

Unsupported Operators

Important operators that are *not* currently supported are the increment and decrement operators (++,– –) and the casting operators (`(int),(double)`). Table 7 lists all of the operators that are not supported.

Table 7. Unsupported Operators

Operators	*Symbols*
Assignment	+=, /=, %=, *=, /=, <<=, >>=, >>>=, &=, ^=, |=
Arithmetic	++, – –
Bitwise	<<, >>, >>>
Boolean	? :
Casting	`(int),(double)`

The following operations are not supported:

- The Boolean ternary operation `c = (a > b) ? a : b;`
- Assignments within an expression such as `while ((a = b-c) > 0)`.

Assignments must be made in a separate statement.

3.5 Program Control Flow

The following constructs are supported:

- `if`
- `for` – Defined loop; number of steps calculated in PSG.
- `while` – Undefined loop; generated real-time expression.

The following constructs are not supported:

- `do..while`
- `switch`

3.6 Function Calls

You can use several types of function calls–such as standard math functions supported by C or Java, string functions, and special functions–to find the maximum or minimum values in a variable. All the function calls return values and you can use them in an expression field.

Math Functions

Supported math functions are listed in Table 8. These functions use the methods that are in the Java Math library. All the math functions return values as doubles. The inputs to the math functions are doubles; however, you can use integers that will be converted to doubles before calling the function.

Because casting doubles to integers is not supported, to round or truncate a double to an integer, use the math functions `rint()` and `floor()`.

Table 8. Supported Math Functions

`double sin(double a);`	Computes the sine of *a*, where *a* is in radians.
`double abs(double a)`	Returns the absolute value of *a* in radians.
`double acos(double a);`	Computes the arc cosine of *a* in radians. The value of *a* should be in the range –1.0 to 1.0.
`double asin(double a);`	Computes the arc sine of *a*. The value of *a* should be in the range –1.0 to 1.0.
`double atan(double a);`	Computes the arc tangent of *a*. The value is returned in the range –pi/2 to pi/2 radians.
`double atan2(double a, double b);`	Converts rectangular coordinates to polar coordinates. The value is returned in the range –pi to pi radians
`double ceil(double a);`	Rounds a number *a* to the smallest integer value greater than or equal to it.
`double cos(double a);`	Computes the sine of a, where *a* is in radians.
`double exp(double a);`	Computes the exponential of *a*, defined as e**a.
`double floor(double a);`	Rounds a number *a* to the largest integer value less than or equal to it.
`double IEEEremainder(double f1, double f2);`	Computes the remainder of the division between two floating point numbers.
`double log(double a);`	Computes the natural log of *a*. The value of *a* must be positive.
`double max(double a, double b)`	Returns the greater of the two numbers.
`double min(double a, double b)`	Returns the smaller of the two numbers.
`double pow(double a, double b);`	Computes the value of *a* raised to the power *b*, a**b. If *a* is negative, *b* must be an integer value.
`double rint(double a);`	Rounds a number *a* to its closest integer value.
`double sqrt(double a);`	Computes the square root of *a*.
`double tan(double a);`	Computes the tangent of *a*, where *a* is in radians.
`double toDegrees(double angrad)`	Converts *angrad* measured in radians to the equivalent angle measured in degrees.
`double toRadians(double angdeg)`	Converts *angdeg* measured in degrees to the equivalent angle measured in radians.

String Functions

You can use the following functions to retrieve information from a string. In all cases where an integer value *n* is requested, you can use a double that will be truncated to an integer when the function is called.

`String getField(String s,int n)`	Returns the *n*th string field from `s`. Fields are separated by spaces, tabs, and new lines.
`String getChar(String s,int n)`	Returns the *n*th string character from `s`. If `n` is greater than the length of the string, the last character is returned.
`boolean yflag(String s)`	Returns a Boolean true or false if the first character of `s` is `y` or `n`.
`boolean yflag(String s,int n)`	Returns a Boolean true or false if the *n*th character in `s` is `y` or `n`. If `n` is greater than the length of the string, the Boolean is based on the last character.

Special Functions

`boolean isArrayed(String var_name)`	Returns a Boolean true if the variable defined by `var_name` is arrayed. Otherwise returns a false.
`double getMaxValue(String var_name)`	Returns the maximum value of the variable defined by the `var_name`. Remember that variables can have a single value or a number of values.
`double getMinValue(String var_name)`	Returns the minimum value of the variable defined by `var_name`. Remember that variables can have a single value or a number of values.

3.7 Special Operations

Outputting Values and Information

To output to standard output, standard error, a file, or VNMR, use the `output` call. The only time to use more than one argument is when a mathematical expression needs to be evaluated before being output.

If an arrayed variable is used, `output` is called for each value.

output Structure

```
output(<output_descriptor>,arg1,arg2,arg3,...,arg1)
```

`output_descriptor` is `stdout`, `stderr`, `vnmr`, or a filename. Arguments can be strings, expressions, values, or variables.

Example 1

The following statement is an example of outputting the values of `pw` to `"stdout"`, where `"stdout"` is currently the text window at the bottom of VNMR that holds the parameter delays:

```
output("stdout","pw = ",+pw+"\n");
```

If `pw` is arrayed and has the values 3.0e-4, 4.0e-4, 5.0e-4, the output is

```
pw = 3/0e-4
pw = 3.999999999998e-4
pw = 5.0e-4
```

Example 2

```
output("stdout","Total time = ",d1+pw+at,"\n");
```

With `d1 = 0.1`, `at = 0.0512` the output is:

```
Total time= 0.1515
Total time = 0.1515999999999998
Total time = 0.1515
```

Example 3:

```
output("vnmr","setvalue('preacq',"+preacq+",'processed')\n")
;
```

With `preacq = 0.01`, this sends the following command to VNMR:

```
setvalue('preacq',0.01, 'processed')
```

Executing Programs

To call executables outside of SpinCAD in a `Calc` field, use a function called `exec`, which is essentially the same as the `psgExec` element.

```
exec(<return_val>,<name>,arg1,arg2,arg3,...,argn)
```

`return_val` is the name of the variable to return output from standard output. If the variable is not a string, any output is converted to the variable type.

`name` is the name of the program to execute. The name can be a full path name or the executable name if the `path` variable has a directory for it.

`arg1-argn` are arguments that are converted to strings, concatenated, and sent as an argument string.

A number of arguments can be used. However, the only time to use more than one argument is when a mathematical expression needs to be evaluated. The `exec` statement works the same way that the `output` statement works. If a variable is arrayed, the program is executed for each value of the array.

The `exec` statement aborts PSG if the exit code of the program is less than zero; otherwise, it is the user's responsibility to check the `return_val` for correct execution.

You can use the `getfield` function to get values out of the `return_val`.

The following example runs the Pbox routine, extracts the pulse width in microseconds, converts the pulse width to seconds, and outputs the value. The pulse width is the second field in the return string.

Example

```
String pboxrtn;
exec(pboxrtn,"Pbox",pwpat+" -w eburp1 -1");
pw = getField(pboxrtn,2);
pw = pw*1e-6;
output("stdout","pw = "+pw+"\n");
```

The output in the VNMR window is

```
pw = 0.0050
```

Chapter 4. Translating C Pulse Sequences to SpinCAD Format

Sections in this chapter:

- 4.1 “vnmr2sc Description and Features” on page 53
- 4.2 “Limitations in vnmr2sc” on page 55
- 4.3 “Using vnmr2sc” on page 57
- 4.4 “Cleaning Up Converted Pulse Sequences” on page 59

There are two ways to convert a pulse sequence from C to SpinCAD.

- Reprogram a sequence “from scratch” or “by hand,” using SpinCAD. This method will lead to sequences that take full advantage of the SpinCAD facilities.
- Use our VNMR C-to-SpinCAD pulse sequence translator, `vnmr2sc`, to directly translate the C pulse sequence into SpinCAD format. This method has the advantage of being faster, but it cannot give perfect results. Sequences that are translated using `vnmr2sc` usually require some cleanup with SpinCAD.

`vnmr2sc` was built to *assist* you in translating VNMR C pulse sequences into SpinCAD format. It is *not* meant to generate perfect SpinCAD pulse sequences. There are fundamental differences between the philosophies and the concepts used in VNMR C and SpinCAD pulse sequences. Even though it might, in many cases, deliver functional SpinCAD pulse sequences, you must use SpinCAD to check and possibly complete or correct the resulting sequence. The current version of `vnmr2sc` does not cover all aspects of the VNMR C pulse sequence syntax; missing features need to be added by hand using SpinCAD.

For detailed information about these limitations, see the section “Limitations in vnmr2sc” on page 55.

Varian, Inc. does not guarantee that sequences converted with `vnmr2sc` are functional or complete.

4.1 vnmr2sc Description and Features

This section provides an outline of what `vnmr2sc` does and how it works.

Construction and Storage of A New Pulse Sequence

`vnmr2sc` is a macro that first calls `dps('-j')`, which produces a `dps` display plus the following two temporary files in the current experiment:

- `dpstable`, which contains the phase information.

- `dpsdata`, which contains the pulse sequence information as used by `dps` for the regular pulse sequence display.

A new pulse sequence in SpinCAD format is constructed from the `dpsdata` and `dpstable` files using a shell script, `/vnmr/bin/vnmr2sc`. The resulting SpinCAD sequence is stored in the local `spincad/psglib` under the same name as the C pulse sequence (the name stored in the `seqfil` parameter), but without the `.c` extension. After the conversion, the files `dpsdata` and `dpstable` are deleted.

`vnmr2sc` requires a SpinCAD element library, `sclib`, to be present either in `/vnmr/spincad` or in your `vnmrsys/spincad` directory. For the conversion of the phase tables, a second utility, `readsctables`, must be present in the command path.

C PSG Functions and Features that Are Converted

The following pulse sequence generation functions and features built into `vnmr2sc` are currently implemented:

- Standard channel definition depending on sequence
- Pulse sequence comment; date in trailer: conversion date/time
- `obspulse`, `pulse`, `rgpulse`, `decpulse`, `decrgpulse`, etc., including `genpulse` and `genrgpulse`
- `simpulse`, `sim3pulse`, `sim4pulse`, including `gensim2pulse`, `gensim3pulse`
- `shaped_pulse`, `decshaped_pulse`, etc., `genshaped_pulse`, `apshaped_pulse` and related functions
- `simshaped_pulse`, `sim3shaped_pulse`, `gensim3shaped_pulse`
- `delay`, `hsdelay` (with properly controlled homospoil pulse)
- Status control on all decoupler channels
- `acquire` (implicit and explicit)
- `zgradpulse`, `rgradient` (Z gradients only)
- `xmtron`, `xmtroff`, `decon`, `decoff`, etc.
- `txphase`, `decphase`, `dec2phase`, etc.
- `xmtrphase`, `dcplrphase`, `dcplr2phase`, etc.
- `recon`, `recoff`, `rcvron`, `rcvroff`
- `obsblankon`, `obsblankoff`, `decblankon`, `decblankoff`, etc.
- `obsoffset`, `decoffset`, `dec2offset`, etc., `offset`
- `obspower`, `decpower`, etc., `rlpower`
- `obspwrf`, `decpwrf`, etc., `rlpwrf`
- `obsprgon`, `obsprgoff`, `decprgon`, `decprgoff`, etc.
- `xgate`
- `sp1on`, `sp1off`, `sp2on`, `sp2off`, `sp_on`, `sp_off`
- Lock sampling control
- `statusdelay`
- `setstatus`
- `incdelay` / `vdelay`
- `spinlock`, `decspinlock`, `dec2spinlock`, etc.

- `loop`, `endloop` (including nested loops), `starthardloop`, `endhardloop`
- `ifzero`, `elsenz`, `endif` (nesting included)
- "`i`" (interactive / `acqi`) elements such as `ipulse`, `ipwrf`, `ipwrm`, etc.

Other Implemented Features

Phase tables up to a maximum length of 1024 steps are extracted from the C pulse sequence, regardless of whether they were defined using external table files, "in-line table syntax" (using `settable` calls), or using real-time math. The length of the phase tables that are extracted depends on the setting of `nt` when `vnmr2sc` is called.

By default, `vnmr2sc` constructs phase tables for all elements requiring phases or real-time variables as arguments. Phase table elements are filled with the phase information as generated by `dps` (at least 32, or up to 1024, elements per table, as specified by `nt`). `vnmr2sc` repetitively divides each table into two halves–and checks whether the two halves are identical–until the table size is an odd number or until the two halves differ. Tables with an odd number of elements cannot be compressed. If the result contains eight or more elements, the tables are further simplified (using SpinCAD shorthand table syntax such as 0∧4 2∧4, where possible) in order to make the result more readable.

The delays `d2`, `d3`, and `d4` in nD experiments are translated to `t1`, `t2`, and `t3`.

Small-angle phase shifts are added to the pulse phases.

`vnmr2sc` detects common pulse width multipliers (for 180°, 270°, and 360° flip angles) and accordingly scales up the visual pulse width for SpinCAD. However, clicking on **Tools** in the SpinCAD main menu then **Re-layout display** resets those pulse widths to the standard or default value!

`vnmr2sc` uses "inside" (rather than purely sequential) docking for AP bus events and gating. Wherever possible, gating and AP bus events are docked inside an adjacent delay. The corresponding timing corrections are removed from the code. This feature also makes the resulting SpinCAD sequence much easier to modify or complete.

`vnmr2sc` considers the differences in the docking and composite layout of rf pulses (compared to their C equivalents): `rof1` or `rof2` timing corrections are removed from delays.

Initial `ifzero` constructs (used only in real-time math) are suppressed (because real-time math is not translated, such constructs would appear empty). There is a very small chance that this feature might suppress a "real" `ifzero`.

4.2 Limitations in vnmr2sc

The current version has a number of limitations that fall into two main categories: inherent limitations and features that have not yet been implemented.

Inherent Limitations

With logical pulse sequence decisions or branchings in C, only one branch is used. In particular, if the sequence of events changes as a function of a variable delay (such as a pulse moving across another pulse), only one case (before or after the crossover) is translated, depending on the current parameter settings.

In general, C calculations, checks, and decisions do not show up in the SpinCAD code.

With “spinlock,” the duration is *not* evaluated and is left blank. In order for the SpinCAD sequence to compile, specify the appropriate parameter or expression in the duration attribute.

Expressions used to initialize (`initval`) real-time variables (such as for loop counters) are lost and need to be manually added.

Autoincrementing tables in C are not properly translated. In SpinCAD, a table index other than `ctss` needs to be constructed for such cases.

`clearapdatatable`, which zeroes all data in the acquisition processor memory, is not shown in `dps` and needs to be manually added.

The low-level functions `G_Delay`, `G_Offset`, `G_Power`, and `G_Pulse` are not shown in `dps` and are therefore not translated.

Loops with built-in real-time math will not result in an appropriate SpinCAD construct.

`vdelay` and `incdelay` result in simple delays with duration attributes (for example, `ph1*1e-3`), which are currently not allowed in SpinCAD and need to be replaced with other suitable constructs, such as a loop containing a simple delay.

Features that Are Currently Not Implemented

The following list includes other functions and features (C pulse sequence functions and features) that are not yet implemented in the current version of SpinCAD or not yet translated by `vnmr2sc`.

Gradients Other Than Z

- `gradient` / `incgradient` / `vgradient`
- `magradient` / `magradpulse`
- `mashapedgradient` / `mashapedgradpulse`
- `obl_gradient` / `oblique_gradient` / `oblique_gradpulse` / `oblique_shapedgradient` / `obl_shapedgradient`
- `shaped2Dgradient` / `shapedgradient` / `shapedincgradient` / `shapedvgradient`
- `vagradient` / `vagradpulse`
- `vashapedgradient` / `vashapedgradpulse`
- `zero_all_gradients`

List Functions

- `create_delay_list` / `create_freq_list` / `create_offset_list`
- `getarray`
- `vdelay_list` / `vfreq` / `voffset`
- `init_decpattern` / `init_gradpattern` / `init_rfpattern` / `init_vscan`

Imaging Functions

- `msloop` / `endmsloop`
- `peloop` / `endpeloop`

- getorientation
- pe_gradient / pe_shapedgradient
- pe2_gradient / pe2_shapedgradient
- pe3_gradient / pe3_shapedgradient
- pe_oblique_shapedgradient / pe_oblshapedgradient
- phase_encode3_gradient / phase_encode3_shapedgradient
- phase_encode_gradient / phase_encode_shapedgradient
- poffset_list / poffset
- position_offset_list / position_offset

Other Functions

hdwshiminit

sli / vsli

rotorperiod / rotorsync

blankingon / blankingoff

readuserap / setuserap / vsetuserap

4.3 Using vnmr2sc

Conditions

In order for vnmr2sc to work, the following conditions must be met:

C Pulse Sequences Must Be Compiled

A C pulse sequence must be compiled **with the dps option** because vnmr2sc uses dps to produce intermediate sequence information. Sequences that cannot be compiled with the dps additions are not translated by vnmr2sc.

What you see in the dps display is what vnmr2sc translates. For example, in the HMQC pulse sequence, the “nulling” of the uncoupled signals is only performed if the delay null has a nonzero value. If vnmr2sc is called with null=0, the BIRD pulse and the delay null are not performed, not shown in dps, and are not included in the resulting SpinCAD pulse sequence. This result applies to any feature that is controlled through a flag parameter (and a decision in C / at go time). Of course, building such decisions into SpinCAD pulse sequences is possible; but because C decisions and other C control structures are not shown in dps, vnmr2sc cannot translate these constructs. To translate sequences with C decisions that are not shown in dps, you have two options:

- Manually add the options not shown in dps with SpinCAD.
- Translate a C pulse sequence into multiple SpinCAD sequences.

Besides the dps restriction, vnmr2sc currently has other limitations. For more information, see the section “Limitations in vnmr2sc” on page 55.

Complete Parameter Set and Configuration of VNMR Needed

In order for `dps` to work, you need a complete parameter set with appropriate parameter settings. Also, VNMR must be properly configured for the pulse sequence (i.e., for the proper number of rf channels, a PFG accessory where required, waveform generators as used by the sequence).

VNMR Needed to Run vnmr2sc

Because `vnmr2sc` uses `dps` output, it can only be called from within VNMR.

Converting a C Sequence

To convert a VNMR C sequence into SpinCAD format, use the following procedure:

1. Compile the sequence (use `seqgen`).
2. Set up the parameters in an experiment as if you wanted to start an acquisition (even on a data station).

   Typically this step involves calling a pulse sequence-specific macro (e.g., `dept` for the DEPT.c sequence).
3. Check the pulse sequence documentation or the C code to determine the maximum phase cycle length in the sequence.

   If the phase cycle is longer than 32 steps, set `nt` to the maximum number of steps before calling `vnmr2sc`. You can specify phase cycles lengths up to 1024. `vnmr2sc` shortens the resulting phase tables, if possible (specify an even number of scans–ideally, a power of 2). Specifying too many scans is not a problem, except that doing so slows down conversion. Phase table/cycle conversion requires VNMR 6.1C or later.
4. Using `dps`, check whether the relevant pulse sequence features are turned on (some sequences permit you to disable certain features by using flag control). If the delays `d2`, `d3`, and `d4` in nD experiments are set to 0 (which is the normal case), then `vnmr2sc` first sets these delays to finite values in order to avoid translating special cases (C constructs) for zero or small evolution time values.

   If setting delay or pulse width parameters to zero turns off sections of a pulse sequence (such as `null=0` with the HMQC pulse sequence), you must set these parameters to a nonzero value, i.e., turn on these optional pulse sequence features, for example:

   ```
   setup('H1','CDCl3') hmqc null=.2 vnmr2sc
   ```

   In this case, you will find a new SpinCAD pulse sequence, `hmqc`, in the local SpinCAD `psglib` (`~/vnmrsys/spincad/psglib`). Because C decisions are hidden by `dps`, they are not translated into SpinCAD. Reconverting a sequence using a different flag or parameter setting is often easier than adding the necessary `if` constructs to an existing SpinCAD sequence. Therefore, select different pulse sequence options and then reconvert the sequence with a different name:

   ```
   null=0 mbond='y' taumb=.1 vnmr2sc('hmbc')
   ```

   If a name is specified, `vnmr2sc` uses that name for the SpinCAD sequence instead of the sequence name in the `seqfil` parameter.
5. It is often *not* necessary to turn on *all* pulse sequence options. For instance, you can easily add transmitter solvent presaturation by dragging “presaturation” composites into an existing pulse sequence (typical C pulse sequence constructs for transmitter

presaturation are somewhat complex and would lead to an unnecessarily complex conversion product).

6. If you want the SpinCAD sequence to address more rf channels than the source sequence, increase the number of rf channels using a numeric argument:

   ```
   vnmr2sc('hmbc',3)
   vnmr2sc(3)
   ```

7. If you want to convert a nongradient sequence into a gradient sequence, specify extra gradient channels using a second numeric argument:

   ```
   vnmr2sc('hmbc',3,1)
   vnmr2sc(0,1)
   ```

   If the first numeric argument is `0`, the number of rf channels is left unchanged.

8. Enter the command `spincad` to run SpinCAD, load the converted sequence, and try saving it. This step accomplishes three things:
   - It points you to any empty attribute field (such as the duration attribute of the "spinlock" composite, which is left blank on purpose because the parametrized duration cannot be extracted from `dps`).
   - It points you to any inconsistency or syntax error.
   - If the sequence syntax is correct, it creates an executable sequence (`*.psg`) in your local `seqlib` directory.

9. Check the converted sequence and try to further simplify it and clean it up (otherwise you might be perpetuating a bad style that has crept into existing C pulse sequences during the past 15 years). See the next section "Cleaning Up Converted Pulse Sequences," on this page.

4.4 Cleaning Up Converted Pulse Sequences

Although converted sequences *might* be functional, they are almost always not coded in the simplest and most straightforward way. Checking any converted sequences is worthwhile and strongly recommended to further simplify the sequence. While checking a converted sequence, keep in mind the conversion limitations listed in section 4.2 "Limitations in vnmr2sc" on page 55.

Checking Converted Sequences

After sequences are converted, make the following checks:

- **Precalculated C Pulse Sequence Function Parameters**–If in the C pulse sequence function parameters are precalculated in separate C statements, manually reimplement these calculations either in a "Calculate" or "Input" element or (better) directly in the SpinCAD attributes that are otherwise using names that do not correspond to VNMR parameter names.
- **Phase Cycle Length Not a Power of 2**–If the phase cycle length is *not* a power of 2, you might have to adjust or correct the phase tables.
- **Calculating Loop Counts**–You might need to manually add expressions used to calculate loop counts.
- **Real-time Math Inside Soft Loops**–Manually implement real-time math inside soft loops by using a suitable SpinCAD construct. The present version of `vnmr2sc` does not handle phase alterations between soft loop cycles.

- **nD experiments**–With nD experiments, set the `t1/t2/t3` coherence selection attribute. Typically, add `fad1/fad2/fad3` to the phase attribute of the same pulse, for example:

  `"ph2"+fad1`

- **Gating**–Some of the gating might require slight docking adjustments. `vnmr2sc` tries docking gating primitives and especially AP bus events into adjacent delays rather than into pulses. But there are cases (such as pulses coded from delays and gating instructions) in which this mechanism can produce unwanted results.
- **Accumulations of Primitives**–If you notice accumulations of primitives, simplify the converted sequence. Check for gradient and rf pulses that were coded from simple delays and gating instructions.

  Primitives are usually docked inside delays rather than sequentially. Therefore, you can remove such elements without (temporarily) tearing apart the entire sequence. After you have ensured that the appropriate power settings are performed through the composites in the sequence, it is not unusual to find that all primitive elements can be removed from a SpinCAD pulse sequence.
- **Receiver Gatings**–Remove unnecessary receiver gatings from UNITY or UNITY*plus* sequences (remember that on the UNITY*INOVA*, the receiver is off by default).
- **Explicit Amplifier Blanking**–Explicit amplifier blanking is unnecessary in most cases because SpinCAD pulses use appropriate prepulse delays (which are automatically subtracted from adjacent delays).
- **Setting Power Level**–Rather than using extra primitives for setting the power level, use the power attributes of the following pulse or “decoupler on” composite to set the power (attenuator and linear modulator) and remove these primitives from the sequence.

CAUTION: `vnmr2sc` **already tries adding the power setting to the relevant pulse or “decoupler on” composites. But you should still be careful when removing primitives that control the power.**

- **Spinlock Loops and Transmitter Presaturation**–Replace spinlock loops and transmitter presaturation constructs by using single SpinCAD composites such as “presaturation” and “MLEV spin lock.” Use these constructs because they dramatically simplify the sequence and reduce handling, compilation, and `go` times.
- **Gradient Pulses**–Simplify coding around gradient pulses; the `gradient` composite has a built-in recovery delay, unlike C, which does not. Typically, you can remove any addition of `grise` to the gradient pulse duration plus any `grise` and `gstab` post-pulse delays.
- **`icosel*` or `*icosel`**–If `icosel*` or `*icosel` was used in a coherence selection gradient pulse, `vnmr2sc` should properly translate this pulse by setting 2D/3D/4D to `t1`. Verify the gradient coherence selection scheme.
- **Unnecessary Timing Corrections**–`vnmr2sc` removes unnecessary timing corrections from delays (because prepulse delays and most AP bus sequence elements are performed inside delays). But, sometimes unnecessary timing corrections are not removed (e.g., before and after a loop or an `if` element). In such cases, manually alter the docking scheme in these places or selectively reintroduce the appropriate timing corrections. Avoid additional timing corrections by taking the following preventive measures:
  - Dock refocusing pulses middle-to-middle to a defocusing and refocusing delay rather than between two delays.

  - Dock other pulses middle-to-end or middle-to-front at the intersection of two adjacent delays rather than having a pulse between delays.
  - With evolution delays from which proper precession terms for the adjacent pulses (such as 2*pw/PI or 4*pw/PI) need to be subtracted, use front-to-end docking.

    In general, modify the result of a conversion so that it looks as simple as possible and is more logical in terms of what the sequence does to the spins. For example, rather than using a delay `tau` that is calculated from a parameter `j`, use `j` directly in the duration attribute of the delays involved.

- **Nonapplicable Elements**–Remove elements that do not apply for the spectrometers supported by SpinCAD.
- **Small-angle Phase Shifting Primitives** –Most small-angle phase shifting primitives can be removed because these phase shifts are added to the phase attribute of any rf pulse.

  Note that small-angle phase shifting in translated sequences resets the 90° phase shift.
- **90° Phase Shifting Elements**–Unless 90° phase shifting elements occur *during* a pulse (i.e., while the rf is on), you can also remove them.
- **Unused Decoupler On/Off Composites**–Removing unused “decoupler on” or “decoupler off” composites from the conversion of “status” makes a sequence both simpler *and* safer. For example, in the DEPT sequence, decoupling really only makes sense during the acquisition, but not during `d1`. Do *not* allow for decoupling while pulses are performed on the decoupler channel.
- **Improving Appearance**–Some adjustments might be required to make the converted pulse sequence look more appealing, for example, switch `if` elements to `true` or `false` mode, as appropriate.
- **Balancing t1 Periods**–If a PI pulse appears in the middle of a t1 evolution period, take care to ensure balanced periods before and after the pulse when t1<rof1. Either ensure that `predelay` is set to zero and no AP translations occur or dock a delay of rof1 into the delay following the pulse.

Appendix. List of SpinCAD Components

This appendix describes the available elements for building a pulse sequence. You can see these elements in the Components toolbox by clicking on the **Elements** menu and choosing some or all of them. Descriptions of pulse elements are in the following sections:

- "Common Attributes" on this page
- "System Collections" on this page
- "Pulse Elements" on this page
- "Primitives" on page 76

Common Attributes

The following common attributes appear several times in the descriptions of elements:

Attributes	
Duration	Delay length; the default unit is sec.
Pulsewidth	Pulse length; for example, `pw`, `pwx`, etc. If specified in absolute values, it must be in sec.
Power	rf attenuator value in dB.
Amplitude	rf modulator value; range 0-4095

System Collections

Clicking on **Show system collections** in the **Elements** menu shows the following collection in the Components toolbox.

Prefix

`prefix` consists of the following phase tables:

Table Label	Value (degrees)	Cycling Index
0	0.0	ctss
90	90.0	ctss
180	180.0	ctss
270	270.0	ctss
0 90	0.0 90.0	ctss
0 180	0.0 180.0	ctss
90 270	90.0 270.0	ctss
0 90 180 270	0.0 90.0 180.0 270.0	ctss
0 180 180 0	0.0 180.0 180.0 0.0	ctss

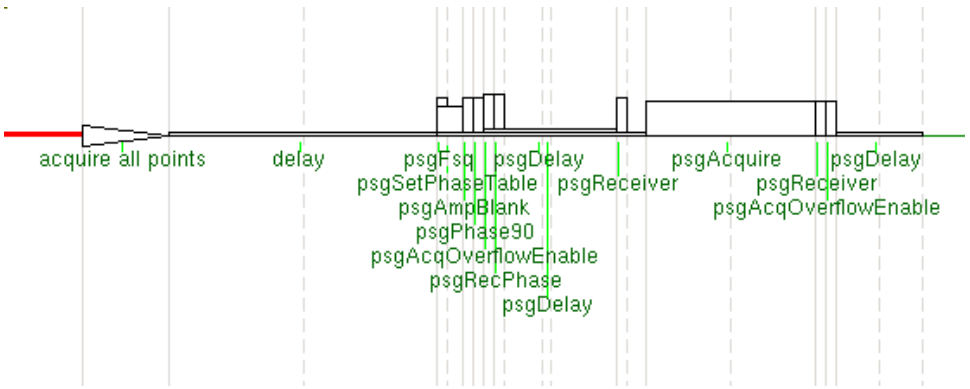
`prefix` is automatically added to the phase table area whenever you click **Sequence** then **New Sequence**.

Pulse Elements

Clicking on **Show pulse elements** in the **Elements** menu shows the following elements in the Components toolbox.

In each element, we have illustrated the appearance of a composite in SpinCAD by docking a delay, “the composite,” a delay, and the composite exploded into primitives.

acquire all points



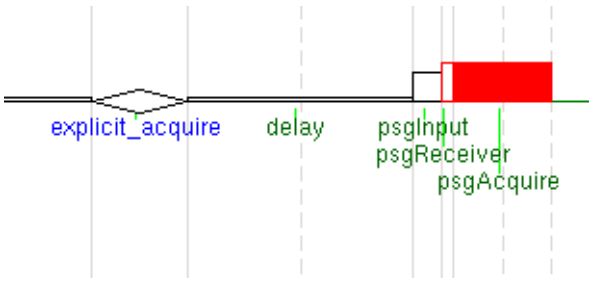
Acquisition element. np/2 complex data points are acquired. The delay before the first data point is digitized is `rof2 + alfa +1/(beta * fb)`.

Attributes

Receiver phase	Phase cycling to be applied to pulse specified in terms of a table name. Select it from the pulldown menu or specify it as the addition/ subtraction of phase tables. `Unaltered`, the default, specifies that the phase is not to be explicitly set but assumes previously set values.
Acq. Quadrature	Select `Cyclops` type phase cycling or indicate the pulse to follow the corresponding setting in the channel. The default is `None`.

acquire (explicit)

Sets up explicit acquisition.

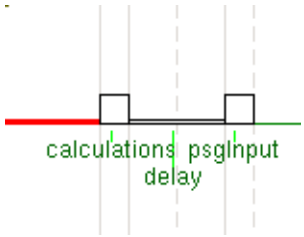


Attributes

Total points	Total number of data points (including real and complex). The default is 2 (one complex point).
Dwell time	Time period between acquisition of data points. The default is 1.0/sw.

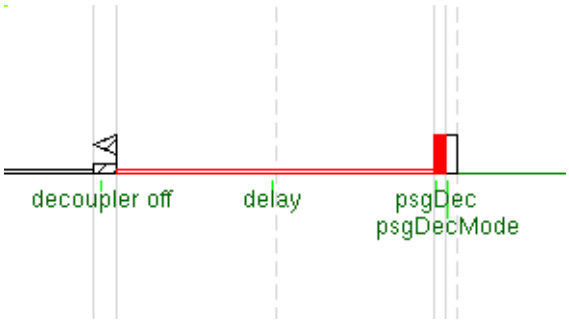
calculations

Provides a place to write calculations separate from any specific attribute in a composite. This distinction is mainly for stylistic purposes.See Chapter 3, “Expressions and Calculations,” for more information.



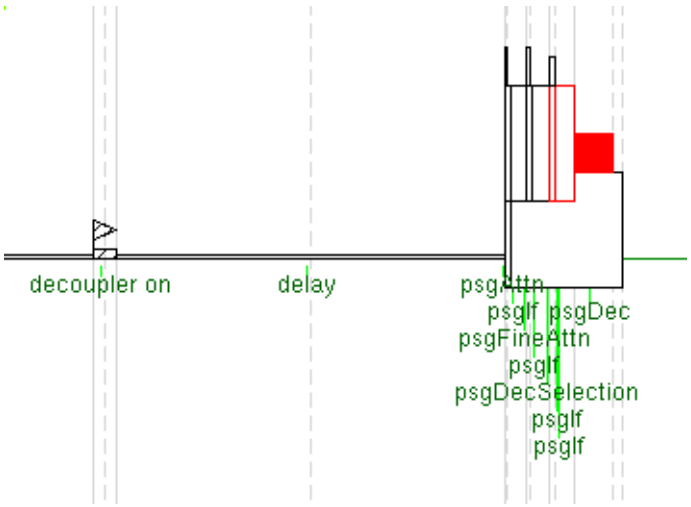
decoupler off

Turns off a decoupler.



decoupler on

Turns on a decoupler.



Turns on various kinds of decoupling schemes–both standard modulations and waveform decoupling (depending on hardware configuration).

Attributes

90 deg. pulse width	The 90° pulse width for a rectangular pulse at the power level used for decoupling (regardless of the shape of decoupling pulses). The value is in sec and is usually 1.0/dmf.
Condition flag	A string or string variable that specifies the condition for turning on the decoupler. It is similar to the dm functionality. The default is `y`.
Character index	Specify the index to be used from a string or string variable for the condition flag, e.g., `3` to select decoupling for the condition flag `nny`. The default is `1`.
Power	Attenuator power in dB; the default is `0`.
Amplitude	Modulator power (0-4095); the default is `Unaltered`.

Mod/shape name	Name of modulation or shape file.
	
	A string or string variable for selecting a standard modulation or shape file (.DEC) names, e.g., “c,” “w,” “m,” or “g” represent cw, WALTZ-16, MLEV, and GARP standard decoupling modes. When a string variable is used, select the relevant characters. For example, to select the second character (“w”) from dmm (e.g., `dmm="cwd"`), use `getChar(dmm,2)`
	
	If a file is not any of the standard modulations, it is presumed to be a waveform-based decoupling using the shape file name (.DEC type). Do not specify “.DEC” in the name.”
	
	Do not specify the `dres` and `dmf` parameters. `dres` and `dmf` calculations are resolved in the following way:
	
	Pulse sequence generation programs attempt to read in `dres` and `dmf` from the specified shape file, from a line of type:
	`#dres="some_number"`
	or
	`#dmf="some_number"`
	
	In both cases, the double quotes are not part of the assignment.
	
	If these keyword patterns are not found in the file, `dmf` is set to 1/(90° pulse width) and `dres` is set to 1. Do not use the VNMR parameters to set these parameters. In some cases, `dres` must be included in the shape file to avoid calculating waveform slices that are too short.

delay

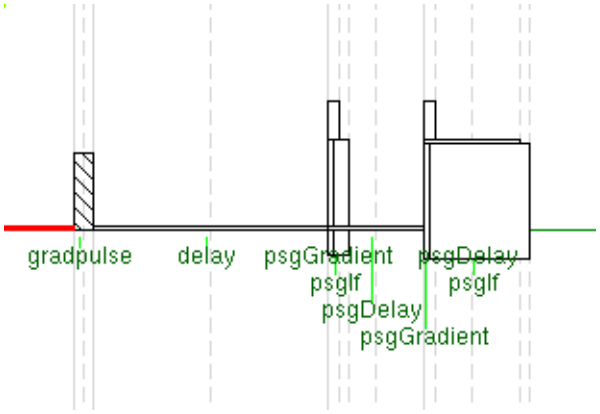
Delay for a specified time.

Attribute

Duration	See the section “Common Attributes” on page 63.

gradient

Applies a gradient pulse.



Attributes

Duration	Length of the gradient in sec.
Amplitude (G/cm)	Strength of the gradient in G/cm. Use the `gcoil` parameter to convert gradient amplitude to a DAC value. See section 2.4 “Gradients Values and Calibration” on page 39 for more information.
Recovery Delay	Delay after the gradient to allow for field recovery in sec. The default is GRADIENT_DELAY.
2D/3D/4D	Select additional amplitude changes (inversion) in the t1, t2, or t3 dimension based on the corresponding phase variable. It is commonly used to implement inversion of gradient amplitude for pure absorption and sensitivity-enhanced gradient experiments. The default is None.

gradient (Oblique)

Applies a gradient along an oblique direction determined by the Euler angles *theta*, *phi*, and *psi*. Figure 47 shows the rotation of the angles, first rotating *psi* (ψ) about the *z* axis, then an angle *theta* (θ) about the *x* axis, and finally an angle *phi* (ϕ) about the z^1 axis.

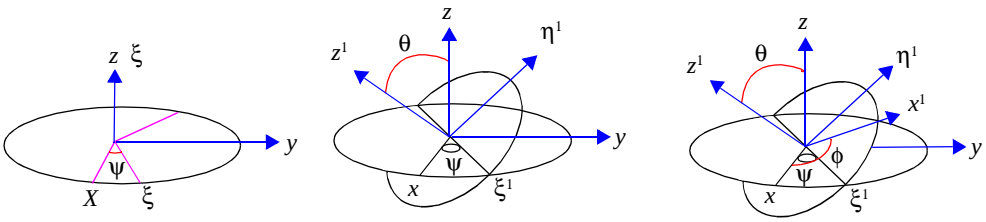
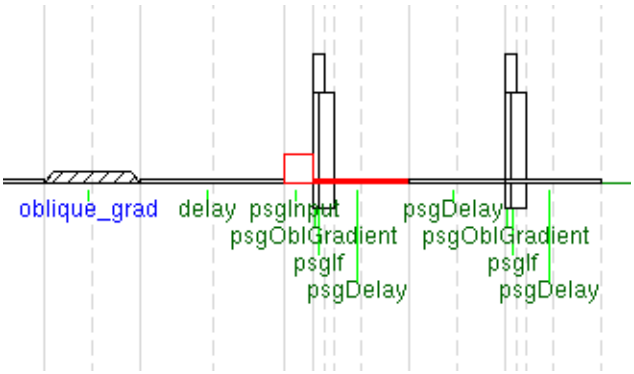


Figure 47. Euler Angles

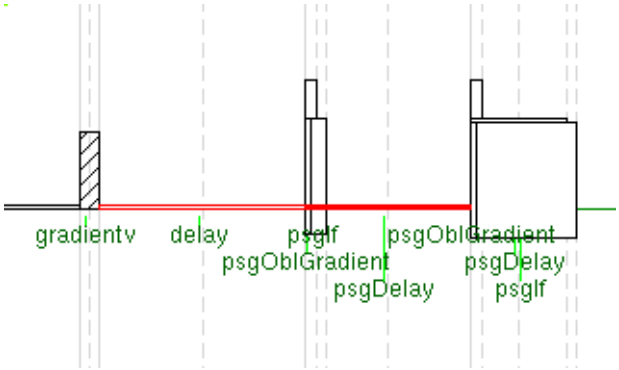


Attributes

Grad Duration	Length of the gradient in sec.
Grad Ampl (G/cm)	Strength of the gradient in G/cm. Use the `gcoil` parameter to convert gradient amplitude to a DAC value. See section 2.4 “Gradients Values and Calibration” on page 39 for more information.
theta	Directional angle *theta* where *theta* is the angle from the *z*-axis and about the *x*-axis.
phi	Directional angle *phi* where *phi* is the angle about the new z^1-axis.
psi	Directional angle *psi* where *psi* is a rotation about the *z*-axis.

gradient (Variable Angle)

Applies a gradient with user-specified orientation in spherical coordinate frame.



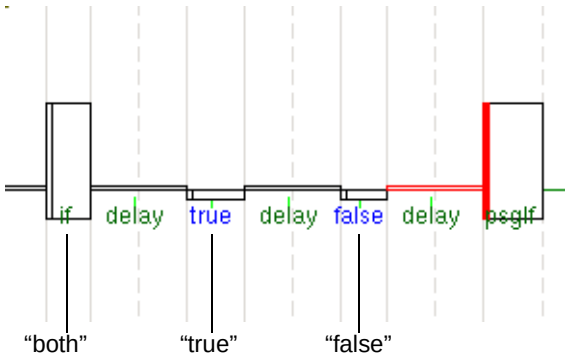
Attributes

Pulse Duration	Length of the gradient in sec.
Amplitude (G/cm)	Strength of the gradient in G/cm. Use the `gcoil` parameter to convert gradient amplitude to a DAC value. See section 2.4 “Gradients Values and Calibration” on page 39 for more information.
theta	Directional angle *theta* for gradient vector where *theta* is the polar angle from the *z*-axis and $0 \le \theta \le \pi$.
phi	Directional angle *phi* for gradient vector where *phi* is the azimuthal angle in the *xy*-plane from the *x*-axis and $0 \le \phi \le 2\pi$.

Recovery Delay	Delay after the gradient to allow for field recovery in sec. The default is GRADIENT_DELAY.
2D/3D/4D	Select additional amplitude changes (inversion) in the t1, t2, or t3 dimension based on the corresponding phase variable. It is commonly used to implement inversion of gradient amplitude for pure absorption and sensitivity-enhanced gradient experiments. The default is None.

if

Conditionally executes pulse elements. The “if” element provides a “true” branch and a “false” branch into which pulse elements can be docked.

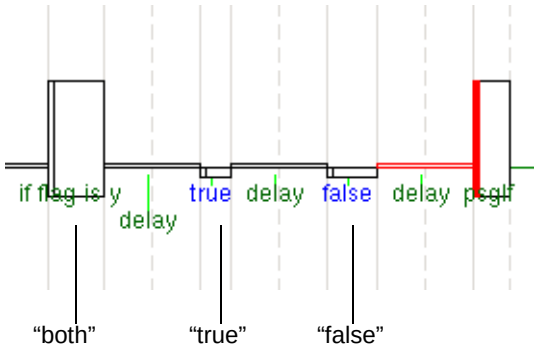


Attributes

Calculations	Calculation field (optional)
Condition	Definition of the condition such as `satmode=="y",tau>1.0`, etc.
Style	Display style. You can select the individual logical branches or both of the branches for viewing from the pull-down menu.

if flag is y

Conditionally executes pulse elements. This element provides a “true” branch and a “false” branch into which pulse elements can be docked. It differs from the “if” element only in the style of declaration of condition.



Attributes

Calculations	Calculation field (optional)

Condition	Definition of the condition flag such as `presat`, `nullflag`, etc. The condition flag is input by itself and not as a relational expression as in the “if” element.
Style	Display style. You can select the individual logical branches or both of the branches from the pull-down menu.

initialize acquisition

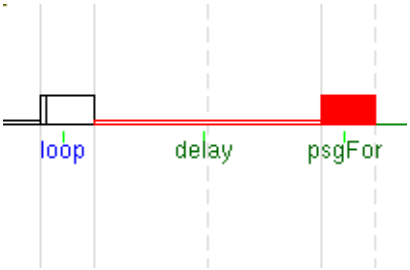
Sets up the various events in the acquisition except for the actual acquisition of the data points. See “acquire all points” on page 64.

Attributes

Receiver phase	Phase cycling to be applied to a pulse specified in terms of a table name. Select it from the pulldown menu or specify it as the addition/subtraction of phase tables. Unaltered, the default, specifies that the phase is not to be explicitly set but assumes previously set values.
Acq. Quadrature	Select cyclops type phase cycling or indicate the pulse to follow the corresponding setting in the channel. The default is None.

loop

For loop, to execute elements multiple times by docking them inside the loop.

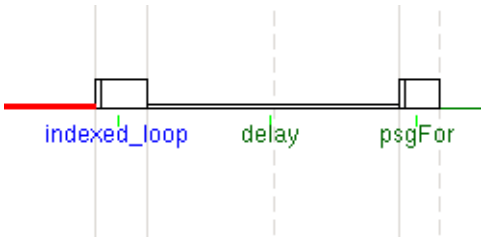


Attribute

Loops	Number of loops to be executed.

loop (with index)

For loop with an index variable available for other calculations.

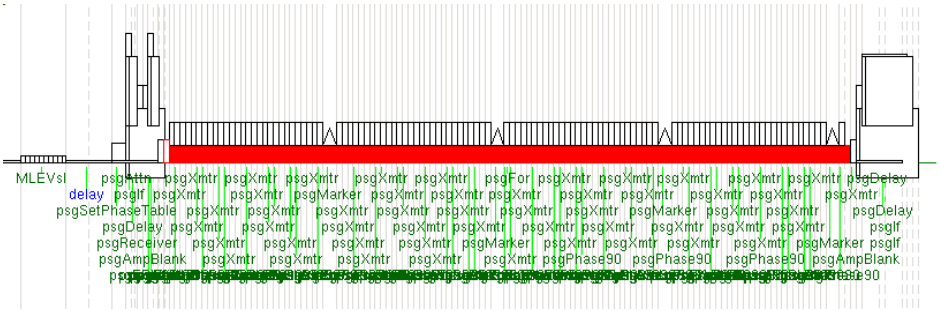


Attributes

Loops	Number of loops to be executed.
Index name	Index variable name that can be used in calculations.

MLEV spin lock

MLEV 17 spinlock composite. It works on any of the RF channels.

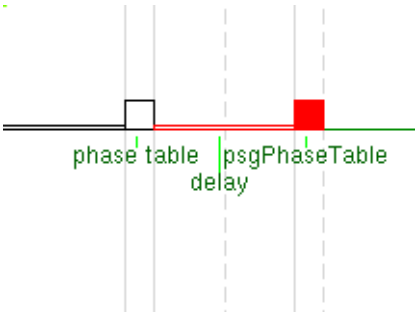


Attributes

Spinlock Duration	Duration (in sec) of spin lock.
Phase Table	Phase cycling to be applied to pulse specified in terms of a table name Select it from the pulldown menu or specify it as the addition/ subtraction of phase tables. Unaltered, the default, specifies that the phase is not to be explicitly set but assumes previously set values.
90deg Pulse width	Pulse width of MLEV pulses (in sec).
Spinlock Power	Attenuator power value (in dB). The default is Unaltered.
17th Pulse angle	Flip angle of 17th pulse (in degrees). The default is 60.0.
Window width	Length of delay between pulses for the clean version (in sec)
Pre/Post Delay	Predelay and postdelay (in sec). The default is 0.0.
Amplitude	Modulator value (0-4095). The default is Unaltered.

phase table

Defines a new phase table.



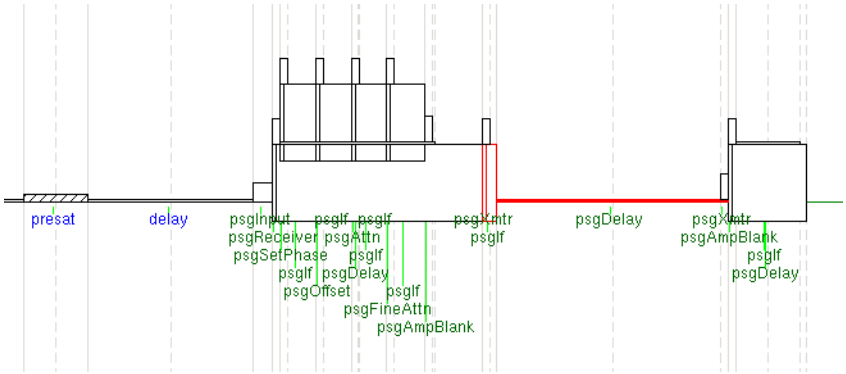
Attributes

Table label	The label or name of the phase table.

Value (degrees)	Phase cycle definition, which is a series of integers representing the phase angle separated by spaces, e.g., 0 270 0 270 180 90 180 90. You can also use a limited syntax of parentheses ( ) and ^ to repeat integers. For example, the definition in the previous example can be entered as (0 270)^2 (180 90)^2. See the section “Phases” on page 34 for more information.
Cycling index	Index to be used for phase cycling, typically ctss (the default). Other indices can also be used.

presaturation

Applies presaturation. This composite can typically be docked end-to-end with the predelay or docked inside an evolution or mixing delay. It works on any of the RF channels.



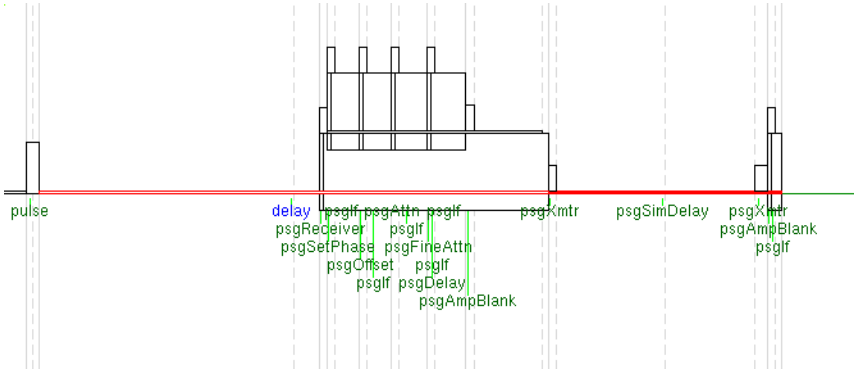
Attributes

Sat. Delay	Length of presaturation (in sec). The default is satdly.
Condition Flag	A string or string variable specifying the condition for applying the presaturation. The default is satmode.
Character Index	Specifies the index of the relevant character to be used from a string or string variable for condition flag (integer). The default is 1.
Sat. Power	Attenuator power value for presaturation (in dB). The default is satpwr.
Sat. Fine Power	Modulator value (0-4095). The default is Unaltered.
Sat. Frequency	Presaturation frequency (in hz). The default is satfrq.
Pre/Post Delay	Predelay and postdelay (in sec). The default is rof1.
Phase Table	Phase cycling to be applied to pulse specified in terms of a table name Select it from the pulldown menu or specify it as the addition/ subtraction of phase tables. Unaltered, the default, specifies that the phase is not to be explicitly set but assumes previously set values. See the section “Phases” on page 34 for more information.
2D/3D/4D	Select additional phase changes to the t1, t2, or t3 dimension based on the corresponding phase variable. It is commonly used to select hypercomplex phase cycling for pure absorption spectra in indirect dimensions. The default is None.
Acq. Quadrature	Select cyclops type phase cycling or indicate the pulse to follow the corresponding setting in the channel. The default is None.

pulse

Applies an rf pulse on the specified rf channel. It works on any of the rf channels.

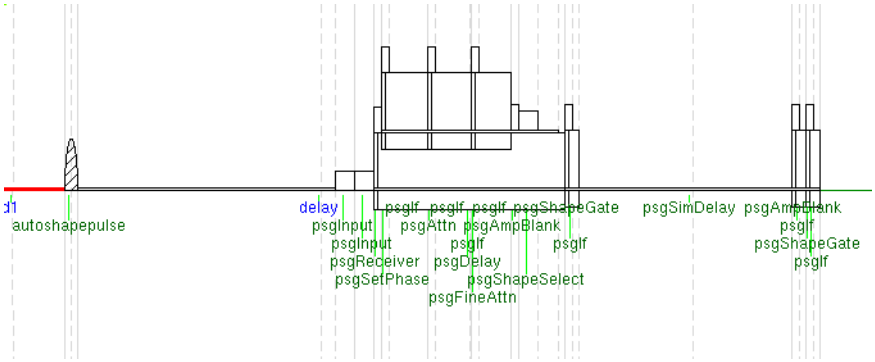
Two or more pulses (shaped or regular) can be simultaneously applied on different channels by docking them middle-to-middle (analogous to applying a simpulse or simshaped pulse in C-psg programming).



Attributes

Pulse width	See the section "Common Attributes" on page 63.
Pre-delay	Pre-delay before the rf transmitter gate is turned on, during which receiver gating, amplifier unblanking, phase setting, etc. takes place. If specified in absolute values, it must be in sec. The default is rof1.
Phase table	Phase cycling to be applied to pulse specified in terms of a table name Select it from the pulldown menu or specify it as the addition/ subtraction of phase tables. Unaltered, the default, specifies that the phase is not to be explicitly set but assumes previously set values. See the section "Phases" on page 34 for more information.
2D/3D/4D	Select additional phase changes to the t1, t2, or t3 dimension based on the corresponding phase variable. It is commonly used to select hypercomplex phase cycling for pure absorption spectra in indirect dimensions. The default is None.
Acq. Quadrature	Select cyclops type phase cycling or indicate the pulse to follow the corresponding setting in the channel. The default is None.
Offset	Synthesizer offset shift. Change the synthesizer setting for this channel to a new value, which produces a synthesizer offset shift during the pre-delay.
Power	See the section "Common Attributes" on page 63.
Amplitude	See the section "Common Attributes" on page 63.

pulse (autoshaped)



Applies an “on-the-fly” shaped rf pulse on the specified channel, after automatically making the shape file using attributes that you specify such as shape type, bandwidth, and/or pulse width. This composite uses the Pbox program to make shape files. Users can optionally preserve these shape files by providing shape file names. This element also provides limited arraying capability. The element works on any rf channel. The shape pulse is automatically calibrated and the attentuator and modulator power values are accordingly set. This element can be docked middle-to-middle to other pulses to perform simultaneous pulses.

Users can array the shape name, pulse width, band width, offset, and phase in a single or multiple array. Reference power and pw90 **cannot** be arrayed.

Attributes

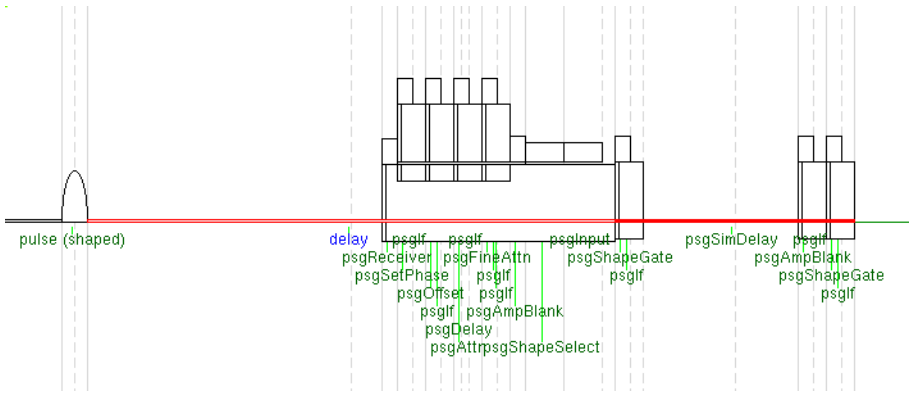
Shape type	Click on the button and select a shape type from the pull-down menu.
Shape name	Select a name using the pull-down menu. You can also set this attribute with a string or string variable. The name must be one of the Pbox shape names.
Pulse width(sec)	Specify pulse width as a constant or a variable. The default unit is sec. You can also use the keyword “Calculate” (the default) if the pulse width is to be computed by Pbox using the specified band width. For some shapes, e.g. adiabatic shapes, you must specify both the pulse width and band width.
Band width(hz)	Specify band width as a constant or a variable. The default unit is hz. You can also use the keyword “Calculate” (the default) if the pulse width is to be computed by Pbox using the specified pulse width. For some shapes, e.g. adiabatic shapes, you must specify both the pulse width and band width.
Frequency Shift (hz)	Shift in frequency (hz) for pulse from the synthesizer frequency for the channel. This frequency shift is implemented by a linear phase ramp in the waveform (SLP) and not by a change in synthesizer frequency. The default is 0.0 hz.
Shape file name	A string or string variable to set the name of the shape file (without the .RF extension). You can also set this attribute to “Auto” (the default) for SpinCAD to automatically make shape names.
Pre-delay	Pre-delay before the rf transmitter gate is turned on, during which time receiver gating, amplifier unblanking, phase setting, etc. occurs. If specified in absolute values, it must be in sec. The default is rof1.

Phase table	Phase cycling to be applied to pulse specified in terms of a table name. Select it from the pulldown menu or specify it as the addition/ subtraction of phase tables. Unaltered, the default, specifies that the phase is not to be explicitly set but assumes previously set values. See the section “Phases” on page 34 for more information.
2D/3D/4D	Select additional phase changes to the t1, t2, or t3 dimension based on the corresponding phase variable. It is commonly used to select hypercomplex phase cycling for pure absorption spectra in indirect dimensions. The default is None.
Acq. Quadrature	Select cyclops type phase cycling or indicate the pulse to follow the corresponding setting in the channel. The default is None.
Reference power (dB)	Value in dB for power/pulse width calculations. e.g., `tpwr` value.
Reference pw90(sec)	Value in sec for pulse width/power calibration, e.g., `pw`. If amplifier compression is to be considered, enter the (compression) corrected pw90.
Addl Pbox options	Enter additional Pbox options (in Pbox format) in this field. Options must be enclosed within double quotes. Multiple options must be separated by a semi-colon **without spaces**, e.g., `"bsim=y;steps=512"` These additional options override any relevant values set in the previous attribute fields. The default is “None”.

pulse (shaped)

Applies a shaped rf pulse on the specified rf channel. This element works on any of the rf channels.

Two or more pulses (shaped or regular) can be simultaneously applied on different channels by docking them middle-to- middle (analogous to applying a simpulse or simshaped pulse in C-psg programming).



Attributes

Pulse width	See the section “Common Attributes” on page 63.
Pre-delay	Pre-delay before the rf transmitter gate is turned on, during which time receiver gating, amplifier unblanking, phase setting, etc. occurs. If specifying in absolute values, it should be in sec. The default is rof1.

Phase table	Phase cycling to be applied to pulse specified in terms of a table name Select it from the pulldown menu or specify it as the addition/ subtraction of phase tables. `Unaltered`, the default, specifies that the phase is not to be explicitly set but assumes previously set values. See the section "Phases" on page 34 for more information.
2D/3D/4D	Select additional phase changes to the t1, t2, or t3 dimension based on the corresponding phase variable. It is commonly used to select hypercomplex phase cycling for pure absorption spectra in indirect dimensions. The default is None.
Acq. Quadrature	Select cyclops type phase cycling or indicate the pulse to follow the corresponding setting in the channel. The default is None.
Offset	Change the offset for this channel to a new value, which produces a synthesizer frequency shift during pre-delay. The default is Unaltered (no change). The shape can optionally contain a linear phase ramp (SLP) to affect additional shift in frequency.
Power	See the section "Common Attributes" on page 63.
Shape amplitude	See the section "Common Attributes" on page 63.

receiver phase

Defines receiver phase cycling.

Attributes

Phase table	Phase table to be used. You can also specify a table as a sum or difference combination of multiple phase tables (all of equal length), e.g., `ph1 = ph2 - ph3` in which `ph1`, `ph2`, and `ph3` are phase tables. You can also select phase tables from a pull-down menu by clicking on the  arrow button next to the blank field. See the section "Phases" on page 34 for more information.
Acq. Quadrature	Select cyclops type phase cycling or indicate the phase is to follow the corresponding channel setting. The default is None.

Primitives

Clicking on **Show primitives** in the **Elements** menu shows the following elements in the Components toolbox.

ACQUIRE

Initiates data acquisition.

Attributes

Duration	Total acquisition time (in sec.).
Total points	The sum of the number of real and imaginary points to be acquired.

AMPLIFIER BLANKING

Blanks or unblanks the amplifier.

Attributes

Duration	The default is 0.0. This event is done using a high speed bit and has no associated duration.
State	On or Off. `On` blanks the amplifier; no RF pulse is output. Off unblanks the amplifier; RF pulse is output.
Set flag	Yes (default) or No. If set to Yes, the status of amplifier blanking can be queried in a calculation window using the `isBlankOnFlag( )` function.

BACKGROUND HARDWARE SHIM

Enables background hardware shimming.

Attributes

Duration	BKGDHWSHIM_DELAY, the AP delay for this element.
Shim Flag	If flag is `y`, shimming can be enabled. The default is the variable hdwshim.
Shim List	List of shims to be adjusted. The default is hdwshimlist.
State	Turn on/off shimming.

CLEAR DATA TABLE

Clears data memory.

Attribute

Duration	The default is 0.0.

DEC STD/SHAPED SELECTION

Selects standard modulation types or waveform-based shapes for decoupling.

Attributes

Duration	DECSELECT_DELAY
Shape name	A string or string variable for selecting a standard modulation or shape file (.DEC) names, e.g., “c,” “w,” “m,” or “g” represent cw, WALTZ-16, MLEV, and GARP standard decoupling modes. Additionally, the entry for this attribute can be a string variable or string constant. See “Mod/shape name” in the section “decoupler on” on page 65 for more information.
Shape width	The 90° pulse length for a rectangular pulse (regardless of decoupling waveform shape) at the same power level that is used for decoupling. The units are in sec.
Shape amplitude	RF modulator value (0-4095).

DECOUPLER FREQUENCY

Sets the decoupler modulation frequency.

Attributes

Duration	DECMODFREQ_DELAY, the AP delay associated with this event.
Modulation frequency	Modulation frequency in hz.

DECOUPLER MODULATION

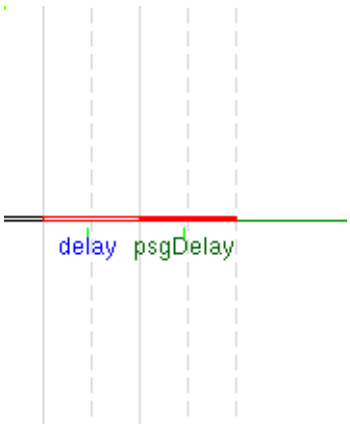
Sets the decoupler modulation mode for standard preprogrammed modulation (nonwaveformed mode).

Attributes

Duration	DECMODMODE_DELAY, the AP delay associated with this event.
Modulation mode	Sets the decoupler modulation from one of the following modes: c Continuous wave (cw) f fm-fm (swept-square wave) g GARP m MLEV-16 r Square wave u User-supplied w WALTZ-16 x xy32

DELAY

Delay for a specified time.



Attribute

Duration	Length of the delay (in sec).

ENABLE OVERFLOW DETECTION

Enables receiver overflow detection.

Attributes

Duration	OVERFLOW_DELAY, the AP delay associated with this event.
State	`On` (default) or `Off`

FSQ MARKER

Used by `psg` to trigger events associated with frequency-shifted quadrature detection.

Attribute

Duration	OFFSET_DELAY, the AP delay associated with this event.

FUNCTION: FOR LOOP

Loop structure to execute elements (docked inside) multiple number of times.

Attributes

User input 1	Attribute fields to define user inputs.
User input 2	Attribute fields to define user inputs.
User input 3	Attribute fields to define user inputs.
User input 4	Attribute fields to define user inputs.
User input 5	Attribute fields to define user inputs.
User input 6	Attribute fields to define user inputs.
Calculations	Calculation field where expressions can be used. Simultaneously press the **Shift** and **Control** keys and click the left mouse button in the attribute field to open the Calculations window. Refer to Chapter 3, “Expressions and Calculations,” for a detailed description of the Calculation field.
Initialization	Initialization statement, e.g., `index=0`.
Test	Testing condition, e.g., `index<=MAX`.
Increment	Increment to the user-defined index, e.g., `index=index+1`.

FUNCTION: IF STATEMENT

Conditionally executes PSG elements. The IF primitive provides a “true” branch and a “false” branch in which PSG elements can be docked.

Attributes

User input 1	Attribute fields to define user inputs.
User input 2	Attribute fields to define user inputs.
User input 3	Attribute fields to define user inputs.
User input 4	Attribute fields to define user inputs.
User input 5	Attribute fields to define user inputs.
User input 6	Attribute fields to define user inputs.
Calculations	Calculation field where expressions can be used. Simultaneously press the **Shift** and **Control** keys and click the left mouse button in the attribute field to open the Calculations window. Refer to Chapter 3, “Expressions and Calculations,” for a detailed description of the Calculation field.
Condition	Logical condition for if, e.g., `satmode=="y"`.
Style	Show individual branches or both branches.

FUNCTION: INPUT

Element to handle user inputs and calculations.

Attributes

User input 1	Attribute fields to define user inputs.
User input 2	Attribute fields to define user inputs.
User input 3	Attribute fields to define user inputs.
User input 4	Attribute fields to define user inputs.
User input 5	Attribute fields to define user inputs.
User input 6	Attribute fields to define user inputs.
Calculations	Calculation field where expressions can be used. Simultaneously press the **Shift** and **Control** keys and click the left mouse button in the attribute field to open the Calculations window. Refer to Chapter 3, "Expressions and Calculations," for a detailed description of the Calculation field.

FUNCTION: RUN PROGRAM

Calls executables outside of SpinCAD. It is essentially the same as the `exec(...)` function that can be executed in a calculation element.

Attributes

Command name	Name of the command to be executed, e.g., `Pbox`.
arguments	Arguments for the command. Several arguments can be used; e.g., `pwpat+"-w eburp -1"`.
return value	String variable, declared by the user, into which the return values are set.

FUNCTION: WHILE STATEMENT

Conditionally executes PSG elements while a condition is true. PSG elements to be executed are docked inside the `while` loop.

Attributes

User input 1	Attribute fields to define user inputs.
User input 2	Attribute fields to define user inputs.
User input 3	Attribute fields to define user inputs.
User input 4	Attribute fields to define user inputs.
User input 5	Attribute fields to define user inputs.
User input 6	Attribute fields to define user inputs.
Calculations	Calculation field where expressions can be used. Simultaneously press the **Shift** and **Control** keys and click the left mouse button in the attribute field to open the Calculations window.
Condition	Logical condition for the while loop, e.g., `index<=MAX`.

GRADIENT

Sets the gradient amplitude values. A gradient can be turned on by setting its value to a nonzero value and turned off by setting the value to zero.

Attributes

Duration	GRADIENT_DELAY, the AP delay associated with this event.
Value	Amplitude value in (G/cm). See section 2.4 “Gradients Values and Calibration” on page 39 for more information.
Polarity (1 or –1)	Only for SpinCAD display purposes. 1 for upright graphic; `-1` for inverted graphic. It **does not** change gradient amplitude or direction.

HOMOSPOIL GRADIENT

Turns on/off homospoil.

Attributes

Duration	HOMOSPOIL_DELAY, the AP delay associated with this event.
State	`On` or `Off`

LOCK SAMPLING

Turns on/off lock sampling.

Attributes

Duration	LOCK_DELAY, the AP delay associated with this event.
Total points	`On` or `Off`

MARKER

Sets visual markers in a pulse sequence. It has no consequence in `psg` execution.

Attribute

Text	Text or comment to be associated with the marker.

OBL GRADIENT

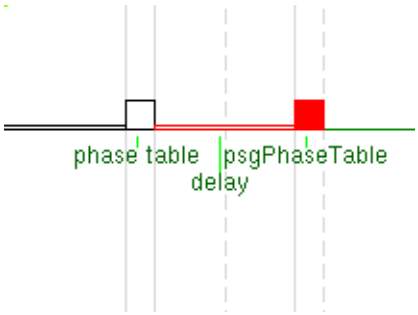
Sets the gradient amplitude and direction expressed in Euler angles. A gradient can be turned on by setting its value and turned off by setting the value to zero.

Attributes

Duration	OBLGRADIENT_DELAY, the AP delay associated with this event.
Value	Amplitude value (in G/cm). See section 2.4 “Gradients Values and Calibration” on page 39 for more information.
Theta	Directional angle theta.
Phi	Directional angle phi.
Psi	Directional angle psi.
Polarity (1 or –1)	Only for SpinCAD display purposes. 1 for upright graphic. `-1` for inverted graphic. It **does not** change gradient amplitude or direction.

PHASE TABLE

Defines a new phase table. Dock this element on the bottom line.

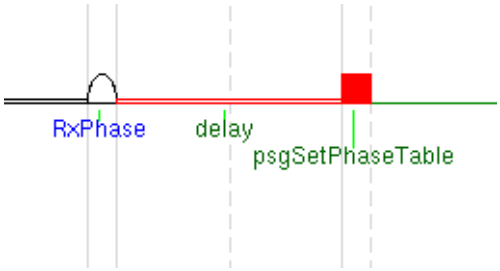


Attributes

Table label	The label or name of the phase table.
Value (degrees)	Phase cycle definition, which is a series of integers representing the phase angle separated by spaces, e.g., 0 270 0 270 180 90 180 90. You can also use a limited syntax of parentheses ( ) and ^ to repeat integers. For example, the definition in the previous example can be entered as (0 270)^2 (180 90)^2. Refer to the section “Phases” on page 34 for a detailed description.
Cycling index	Index to be used for phase cycling. Other indices can also be used. The default is `ctss`.

PHASE TABLE EDIT

Defines a new phase table.



Attributes

Duration	The default is `0.0`.
Table label	The label or name of the phase table.
Value (degrees)	Phase cycle definition, which is a series of integers representing the phase angle separated by spaces, e.g., 0 270 0 270 180 90 180 90. You can also use a limited syntax of parentheses ( ) and ^ to repeat integers. For example, the definition in the previous example can be entered as (0 270)^2 (180 90)^2. Refer to the section “Phases” on page 34 for a detailed description.

Cycling index	Index to be used for phase cycling. Other indices can also be used. The default is `ctss`.
Acq. Quadrature	Select Cyclops type phase cycling or indicate the pulse to follow the corresponding setting in the channel. The default is `None`.
Evolution dimension	Select additional phase changes in the t1, t2, or t3 dimension based on the corresponding phase variable. The default is `None`.

RECEIVER GATE

Turns on/off receiver gate.

Attributes

Duration	The default is `0.0`.
State	On or Off
Set flag	Sets the receiver gate. The default is Yes (default).

RECEIVER HW PHASE

Sets the receiver hardware using a phase table.

Attributes

Duration	RCVRPHASE_DELAY, the AP delay associated with this primitive.
Value	Phase table name.

RF: FINE POWER

Sets the RF modulator value.

Duration	PWRF_DELAY, the AP delay associated with this event.
Value	Modulator value (0-4095)

RF: GATE

Turns on/off the transmitter gate.

Attributes

Duration	The default is 0.0 (high-speed line)
State	On or Off

RF: OFFSET

Sets the RF synthesizer offset.

Attributes

Duration	OFFSET_DELAY, the AP delay associated with this event.
Value	Frequency in hz.

RF: PHASE

Sets the RF phase including the small angle phase.

Attributes

Duration	SAPS_DELAY, the AP delay associated with this event.
Value	Phase value in degrees.

RF: PHASE [90 DEG]

Sets the RF phase (90° part and mulitples).

Attributes

Duration	The default is 0.0 (high-speed line).
Value	Phase value in degrees.

RF: PHASE [FROM TABLE]

Sets the RF phase table (including the small angle part) from a table.

Attributes

Duration	SAPS_DELAY, the AP delay associated with this event.
Value (degrees)	Table name or phase list. Additions and/or subtractions of table names and the phase list can also be specified.
Cycling index	The index is to be used for phase cycling. The default is ctss. Other indices can also be used.
Evolution dimension	Can be used to select additional phase changes corresponding to selected t1, t2, or t3 dimension time incrementation. It is commonly used to select hypercomplex phase cycling for pure absorption spectra in indirect dimensions. The default is None.
Acq. Quadrature	Can be used to add in cyclops-type phase cycling or to follow the corresponding setting in the channel. The default is None.

RF: POWER

Sets the RF attenuator value.

Attributes

Duration	POWER_DELAY, the AP delay associated with this event.
Value	Attenuator value (–16→ +63) dB.

SHAPE: GATE

Turn on/off waveform shape gate.

Attributes

Duration	WFG_OFFSET_DELAY, the AP delay associated with this event.
State	`On` or `Off`

SHAPE: GRADIENT SELECT

Sets waveform shape for a gradient.

Attributes

Duration	WFG_START_DELAY, the AP delay associated with this event.
Shape name	Shape waveform file name (.GRD type)
Shape width	Duration of shaped gradient event (in sec).
Shape amplitude	Amplitude value for gradient.

SHAPE: SELECT

Select shape for a shaped RF pulse event.

Attributes

Duration	WFG_START_DELAY, the AP delay associated with this event.
Shape name	Shape waveform file name (.RF type). For systems without waveform boards, pulse shaping is done using the AP bus.
Shape width	Duration of shaped RF pulse event (in sec).
Shape amplitude	Amplitude value for RF shape amplitude scaling.

SIMULTANEOUS DELAY

Delay for a specified time. Furthermore, if two simultaneous delays are docked middle-to-middle, the elements docked to the front and end of each of delay will happen before or after the longer of the two simultaneous delays. **Do not use this element for typical delays in pulse sequences.**

Attribute

Duration	Length of simultaneous delay (in sec).

SPARE TTL LINES

Triggers on/off spare TTL lines.

Attributes

Duration	The default is 0.0 (high-speed bit).
Spare line	Spare line number (`1-5`).
State	`On` or `Off`

TABLE

Defines a general table to hold a list of values.

Attributes

Table name	Name of the table.
Table values	List of values, separated by a space.
Table index	Index to be used for accessing table entries, e.g., ctss or any other integer.

XGATE:

External trigger gate. Suspends the pulse sequence. When the number of external events has occurred, the pulse sequence continues.

Attributes

Value	The number of external trigger events after which a pulse sequence continues.

A

acquire pulse element, 76
acquiring
  data, 76
  data points, 64
acquisition
  element, 64
  initializing, 70
adding a collection of pulse elements, 41
amplifier blanking pulse element, 77
applying pulse elements
  "on-the-fly" shaped rf pulse, 74
  gradient, 68
  gradient along an oblique direction, 67
  gradient pulse, 67
  presaturation, 72
  rf pulse, 73
  shaped rf pulse, 75
assignments, 48
attributes
  element, printing, 24
  pulse element, 30
  specifying channel, 29
audio/visual control, 24
autoshaped pulse, applying an, 74

B

background
  changing color of drawing canvas, 24
  hardware shim pulse element, 77
  hardware shimming, enabling, 77
blanking amplifier, 77
Boolean operation, unsupported, 48
building pulse sequences, 13, 29, 63

C

C math operators, 47
C pulse sequence
  converting into SpinCAD format, 58
  translating a, 53
Calc field
  performing calculations in the, 46
  running programs in a, 51
calculations
  and expressions, 43
  writing, 65
calibration, 39
calling executables outside of SpinCAD, 80
canvas
  changing layout, 26
  magnifying view of the, 26
  printing the, 25
casting operation, 48
changing
  canvas layout, 26
  channel labels, 11
  colors, display, 24
  docking settings, 15
channels
  attributes, specifying, 29
  indicating number of, 28
  lines, changing color of, 24
clear data table pulse element, 77
clearing data memory, 77
collections
  adding, renaming, or deleting a, 41
  making pulse sequence, 41
common attributes, pulse elements, 63
components, showing, 22
COMPVALUE construct, 46
conditional execution of pulse elements, 69
conditions for running vnmr2sc, 57
Configure menu, 23
connecting elements, 32
connection lines, changing color of, 24
constructs
  COMPVALUE, 46
  supported and unsupported, 48
controlling
  audio/visual preferences, 24
  color, 24
  sound, 24
converted SpinCAD pulse sequences, checking, 59
copying elements, 31
correcting converted SpinCAD pulse sequences, 59
creating multiple channels, 11, 28
customizing the display, 24

D

d0 parameter, 41
data
  acquiring, 76
  memory, clearing, 77
  points, acquiring, 64
  printing, 24
DEC STD/shaped selection pulse element, 77
declarations, variable, 46
decoupler
  frequency pulse element, 78
  modulation pulse element, 78
  turning on/off, 65, 66
decoupler modulation
  frequency, setting, 78
  mode, setting, 78
decoupling
  selecting modulation types for, 77
  turning on/off waveform, 66
decrement operator, 48
defining
  a general table, 85
  a new phase table, 71, 82
  pulse element attributes, 30
  receiver phase cycling, 76
delay, 66, 78
  pulse element, 78
  simulataneous, 85
deleting a collection of pulse elements, 41
disconnecting pulse elements, 32
do..while construct, 48
docking
  elements, 31, 32
  lines, hiding, 24
  order, changing, 15
  points, setting, 32

pulse elements, 15
sound, turning on/off, 24
double operator, 48
drawing canvas, 31
customizing the, 24
duplicating an element, 31

E

elements
components, showing, 22
docking, 31, 32
duplicating, 31
printing descriptions of, 24
stacking, 32
Elements menu, 22, 63
enable overflow detection pulse element, 78
enabling
background hardware shimming, 77
receiver flow detection, 78
Euler angles, 67
events, setting up in acquisition, 70
exec function, 51
executables outside of SpinCAD, calling, 80
executing
programs, in a CALC field, 51
PSG elements conditionally, 79, 80
pulse elements conditionally, 69
SpinCAD, 10
explicit acquisition, setting up, 64
expressions
and calculations, 43
operators, 47
programming, 43
external trigger gate, 86

F

for construct, 48
frequency-shifted quadrature, triggering events for detection, 79
FSQ marker pulse element, 79
function calls, 48
functions, 51
for loop pulse element, 79
if statement pulse element, 79
input pulse element, 80
run program pulse element, 80
while statement pulse element, 80

G

general table, defining a, 85
gradient
amplitude and direction, setting, 81
amplitude values, setting, 81
applying a, 67, 68
pulse element, 81
pulse, applying a, 67
values, 39
graphic canvas, printing the, 25

H

hiding docking lines, 24
homospoil
gate pulse element, 81
turning on/off, 81

I

if construct, 48
increment operator, 48
initializing acquisition, 70
initiating data acquisition, 76
int operator, 48

L

labels, 27
changing Channel, 11
lock sampling
pulse element, 81
turning on/off, 81
looping, 70, 79

M

magnifying the drawing canvas, 26
main menu, 21
mapping virtual channels, 19
marker pulse element, 81
math
functions, 48
operators, C, 47
memory, clearing data, 77
menu items, seeing descriptions of, 24
menus, 21–27
elements, 22
label, 27
main, 21
permanently opening, 22
sequence, 21
tools, 23
MLEV 17 spinlock sequence composite, 71
modulation types, selecting for decoupling, 77
modulations, turning on/off standard, 66
multiple channels, creating and naming, 11

N

naming
channels, 11, 29
pulse sequences, 28
new pulse sequence
constructing and saving a, 54
creating a, 10

O

OBL gradient pulse element, 81
on-the-fly shaped rf pulse, applying, 74
operations, special, 50
operators
casting, 48

expressions, 47
increment and decrement, 48
supported and unsupported math, 47
outputting information, 50
overhead delay time, setting, 41

P

permanently opening menus, 22
phase table
constructed by vnmr2sc, 55
defining a new, 71, 82
edit pulse element, 82
pulse element, 82
presaturation, applying, 72
primitive pulse elements, 76
printing, 24
program control flow, 48
programming expressions, 43–51
programs, executing in a Calc field, 51
PSG
Audio/Visual Control window, 24
elements, conditionally executing, 79, 80
functions in vnmr2sc, 54
pulse elements, 63
adding, renaming, or deleting a collection of, 41
changing color of, 24
conditionally executing, 69
docking, 15
removing, 32
pulse sequences
building, 13
collection, making a, 41
creating, 10
field, 27
naming, 28
running, 19
saving, 19
suspending, 86
translator, 53

R

receiver
gate pulse element, 83
gate, turning on/off, 83
hardware phase pulse element, 83
hardware, setting, 83
overflow detection, enabling, 78
phase cycling definition, 76
removing
a collection of pulse elements, 41
a subset of elements, 32
renaming
channels, 11
collection, pulse elements, 41
RF
attenuator value, setting, 84
fine power pulse element, 83
gate pulse element, 83
modulator value, setting, 83
offset pulse element, 83
phase 90 deg pulse element, 84
phase from table pulse element, 84
phase pulse element, 84
phase table, setting, 84
phase, setting, 84
power pulse element, 84
pulse, applying a shaped, 75
pulse, applying an, 73
synthesizer offset, setting, 83

S

safety limits, channel, specifying, 29
saving a pulse sequence, 19
seeing menu item descriptions, 24
selecting shape for a RF pulse event, 85
Sequence menu, 22
sequences with C decisions, translating, 57
setting
decoupler modulation frequency, 78
docking points, 32
gradient amplitude and direction, 81
gradient amplitude values, 81
overhead time delay, 41
receiver hardware, 83
RF attenuator value, 84
RF modulator value, 83
RF phase, 84
RF phase table, 84
RF synthesizer offset, 83
visual markers, 81
waveform shape, 85
setting up
explicit acquisition, 64
initial acquisition, 70
shape
gate pulse element, 84
gradient select pulse element, 85
select pulse element, 85
selecting for a shaped RF pulse event, 85
shaped rf pulse, applying a, 75
shimming, enabling background hardware, 77
showing/hiding
docking lines, 24
labels, 27
tool tips, 24
simultaneous delay pulse element, 85
sound, controlling, 24
spare TTL lines
pulse element, 85
triggering on/off, 85
special functions, 50
special operations, 50
specifying channel attributes, 29
SpinCAD format, building a pulse sequence in, 54
stacking elements, 32
standard modulations, turning on, 66
starting SpinCAD, 10
string functions, 50
supported
constructs, 48
operations, math, 47
suspending a pulse sequence, 86
switch construct, 48
system collections, pulse elements, 63

Index

T

table pulse element, 85
table, defining a general, 85
timing corrections, avoiding, 60
tool tips, showing/hiding, 24
toolbox, 29
transmitter gate, turning on/off, 83
TTL lines, triggering spare, 85
turning on/off
    decoupler, 65, 66
    docking sound, 24
    lock sampling, 81
    receiver gate, 83
    standard modulations, 66
    transmitter gate, 83
    waveform decoupling, 66
    waveform shape gate, 84

U

unblanking amplifier, 77
undocking pulse elements, 32
unsupported
    constructs, 48
    operations, math, 48
updating labels, 27

V

values
    gradient, 39
    outputting, 50
variables, declarations, 46
virtual channels, mapping, 19
visual markers, setting, 81
VNMR parameters, 46
vnmr2sc
    limitations, 53
    macro, 53
    requirements, 54
    using, 57
volume control, 24

W

waveform decoupling, turning on, 66
waveform shape
    gate, turning on/off, 84
    setting for a gradient, 85
waveform-based shapes, selecting for decoupling, 77
while construct, 48
writing calculations, 65

X

XGATE pulse element, 86

Z

zooming in on the drawing canvas, 26