OXFORD INSTRUMENTS

MARAN Ultra Pulse Programming and Pulse Sequence User Manual

MARAN Pulse Sequence User Manual (for RINMR software version 4.0 or later)
v1.1 April 2006

Note: It is recommended that users access the CD-ROM supplied with the instrument for the latest version of this manual.

Please report comments, errors and omissions to:

Customer Support
Oxford Instruments Molecular Biotools Ltd.
Tubney Woods, Abingdon
Oxfordshire
OX13 5QX
United Kingdom
Tel: +44 (0) 1865 393311
Fax: +44 (0) 1865 393333
E-mail. helpdesk.nanoscience@oxinst.co.uk

Chapter 1 **Introduction** 7

Chapter 2 **Pulse Programming Using MARAN Internal Hardware** 8

2.1 Introduction 8

2.2 Specifying the Default Editor and Compiling the Pulse Sequence 8

2.3 Pulse Program Skeleton 8

2.4 Global Pulse Sequence Variables 8

2.5 OIMBL Pulse Programming Conventions 9

2.6 The Duration Procedure 9

2.7 Local Pulse Sequence Variables 10

2.8 Generating RF Pulses and Phase Cycling 11

2.9 Data Acquisition 12

2.10 A Simple FID Sequence 13

2.11 Controlling Gradients 14

2.12 A Simple Gradient Echo Sequence 15

Chapter 3 **Pulse Programming Using MARAN Ultra Hardware** 17

3.1 Introduction 17

3.2 Specifying the Default Editor and Compiling the Pulse Sequence 17

3.3 Pulse Program Skeleton 17

3.4 Global Pulse Sequence Variables 17

3.5 OIMBL Pulse Programming Conventions 18

3.6 Information Files 18

3.7 The Duration Procedure 19

3.8 Local Pulse Sequence Variables 20

3.9 Generating RF Pulses and Phase Cycling 20

3.10 Data Acquisition 22

3.11 A Simple FID Sequence 23

3.12 Fast RF Pulse Generation 23

3.13 Multi Channel RF Pulse Sequences 23

3.14 Controlling Gradients 25

3.15 A Simple Gradient Echo Sequence 26

3.16 Incrementing Gradients 27

3.17 Shaped RF Events 28

3.18 Shaped Gradient Events 30

3.19 Multi-Dimensional Data Acquisition and the Loop Command 31

3.20 Varying the Pulse Amplitude within a Sequence for Spin Locking 33

Chapter 4 Library of Pulse Sequences and automated Scripts 35

4.1 Start Up Parameters 35

4.2 Pulse Sequences & Scripts 35

4.3 FID Pulse Sequence 36

4.4 FIDRF2 Pulse Sequence 37

4.5 FIDHCW Pulse Sequence 38

4.6 CPMG Pulse Sequence 39

4.7 CPMGGRAD Pulse Sequence 40

4.8 CPMAS Pulse Sequence 42

4.9 INVREC Pulse Sequence 43

4.10 HAHN Pulse Sequence 44

4.11 FID_HAHN Pulse Sequence 45

4.12 FIDCPMG Pulse Sequence 46

4.13 CPMGF Pulse Sequence 47

4.14 SOLID Pulse Sequence 48

4.15 PROFILE Pulse Sequence 49

4.16 DEGAUSS Script 50

4.17 WOBBLE Pulse Sequence 50
4.18 DIFF Pulse Sequence 51
4.19 DIFFA Pulse Sequence 52
4.20 GETDIFF Script 54
4.21 GETDIFFA Script 54
4.22 PGSEMASS Pulse Sequence 54
4.23 GETPGSEMASS script 55
4.24 DROPLET Pulse Sequence 56
4.25 PGSEMASS Pulse Sequence 57
4.26 GETPGSEMASS script 59
4.27 T1 Script 59
4.28 GETDROP Script 60
4.29 DSD Script 60
4.30 TRAIN90 Pulse Sequence 60
4.31 TRAIN180 Pulse Sequence 61
4.32 FID_ONE Pulse Sequence 61
4.33 LISTGEN Script 62
4.34 PROBECHANGE Script 63
4.35 GRADCALIB Script 63
4.36 SETT1RHO Pulse Sequence 64
4.37 T1RHO Pulse Sequence 65
4.38 GETT1RHO Script 66
4.39 TD Script 66
4.40 Imaging sequences and scripts 66

Chapter 5 The RINMR File Format 67

5.1 Introduction 67

5.2 RINMR File Format Definition 67

5.3 Header 67

5.4 Data Files 67

5.5 Image Files 68

5.6 Pascal Header Definition 68

5.7 IDL Image Data Loader 72

Chapter 1 Introduction

The RINMR Programming Manual includes:

- A tutorial including functions and commands to create pulse sequences for RINMR (Chapters 2 and 3).
- A description of the library of pulse sequences and automated scripts supplied with RINMR (Chapter 4).
- The data format for *.RiDat and *.RIImage files (Chapter 5).

A general overview of the RINMR software, including information on loading and running pulse sequences and scripts may be found in the RINMR User Manual. Programmers should note that RINMR may run on three different hardware types and that the pulse programming language for each hardware type is different. The three types of hardware are:

- MARAN Internal Hardware. The pulse programming language for MARAN internal hardware systems is described in Chapter 2.
- MARAN Ultra Hardware. The pulse programming language for MARAN Ultra benchtop, imaging and spectroscopy systems is described in Chapter 3.

If you are in doubt with the programming language applicable to your system, please contact OIMBL for advice.

Chapter 2 Pulse Programming Using MARAN Internal Hardware

2.1 Introduction
The set of procedures and functions that define the events and timing of an NMR experiment are known as a pulse sequence or pulse program. A typical pulse sequence specifies a number of RF pulses, phases, timing durations and acquisition delays. When the pulse program is compiled, the pulse sequence is loaded into the pulse sequence generator for execution.

2.2 Specifying the Default Editor and Compiling the Pulse Sequence
The default pulse sequence editor must be specified in the RINMR software before pulse sequence editing and compilation can take place. The RINMR Developers Kit is supplied with Borland Turbo Pascal 7.0 as standard for pulse sequence compilation. Users should ensure that the correct path (usually c:\tp\bin\turbo.exe) and correct default pulse sequence directory (usually c:\program files\resonance\RINMR\seq\source) are specified under the Tools menu of the RINMR software.

2.3 Pulse Program Skeleton
The pulse sequence must be defined as a Pascal procedure called Sequence. The following listing can be used as a skeleton and is stored on disk in the c:\program files\resonance\RINMR\seq\source directory:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}

USES
{$I UNITS95.INC}

VAR
{Define Global variables here}

PROCEDURE Sequence;

BEGIN
{Enter pulse sequence here}
END;

BEGIN
Run(Sequence);
END.
```

Note that the unit UNITS95.INC must be specified under the USES list and that the three commands at the end of the sequence MUST be present to tell the compiler to run the sequence.

2.4 Global Pulse Sequence Variables
A set of global, pre-defined variables exists for use in the pulse sequences. All the pre-defined variables may be specified using the RINMR data acquisition software.

RINMR Parameter Name	Intended Use	Pulse Programming Parameter Name
P90	90 degree pulse length	P90
P180	180 degree pulse length	P180
SI	Size of acquisition buffer	Points
TAU	Duration	Tau
DEAD1	Probe dead time	Dead1

DEAD2	Filter settle time	Dead2
RD	Experiment recycle delay	RD
DW	Sample frequency	Dwell
NS	Number of scans	Scans
PH x (x=1-5)	Phase program	PH x
D x (x=1-9)	Delays	D x
P x (x=1-9)	Pulses	P x
C x (x=1-5)	Counters	C x
FW	Filter width	FW
NECH	Number of echoes	NumEchoes
G x (x=1-9)	Gradient amplitudes	G x
GX	X gradient global scalar	GX
GY	Y gradient global scalar	GY
GZ	Z gradient global scalar	GZ

Users should note that many of the global variables are accessed using different names in the pulse programming language than from the RINMR acquisition software (for example, to access SI, the Points variable should be used in the pulse program).

2.5 OIMBL Pulse Programming Conventions

Pulse programs written by OIMBL follow a convention to facilitate the operation of pulse sequences by users. It is recommended that pulse sequences written by programmers adhere to this convention.

Phase Programs

The first event (which is nearly always an RF pulse) is assigned phase list PH1.
The ADC phase list is defined as PH2.
The second event phase list is defined as PH3.
The third event phase list is defined as PH4 etc.

Delays

Tau is used as the time delay between consecutive RF pulse centres (for example in the CPMG pulse sequence).

In other pulse sequences where time duration is specified from the end of one RF pulse to the beginning of another, a Dn variable is used for the duration (for example the SOLID pulse sequence, CPMAS sequence).

RF Pulses

P90 is used where a 90° pulse is required.
P180 is used where an 180° pulse is required.

Gradients

The first gradient event is assigned amplitude G1.
The second gradient event is assigned amplitude G2 etc.

ZeroTime

ZeroTime is defined as the start of the first RF event.

2.6 The Duration Procedure

Pulse programs are comprised of a series of Duration procedures. The Duration procedure takes two arguments. The first argument is a variable of type REAL or a constant that defines the length in time of the Duration procedure. The second argument (action argument/function) tells the hardware what actions to perform during the Duration procedure.

For example:
```
Duration(10*US,0);
```

This command instructs the pulse sequencer to do nothing for 10 microseconds. In addition to the constant, a time multiplier (either US, MS or S) must be specified.

The pre-defined variables may be specified as arguments to the Duration procedure.
For example:
```
Duration(D1*US,0);
```

This command instructs the pulse sequencer to do nothing for D1 microseconds. D1 may be altered interactively from within RINMR. The different types of action arguments/functions that may be passed to the Duration command are listed in the following table:

Argument	Description	Example
0	Do nothing	Duration(D1*US,0)
RF(PHn)	Output on the first RF channel, use phase program PHn	Duration(P90*US,RF(PH1))
ADC(PHn)	Acquire one pair of points (simultaneously), use phase program PHn	Duration (DW*US,ADC(PH2))
REC	Enable the receiver	Duration(DEAD2*US,REC)

More than one action argument/function may be passed using the addition operator, for example:

```
Duration(D1*US,REC+ADC);
```

More information on the use of specific action arguments is contained in sections 2.9-2.13.

2.7 Local Pulse Sequence Variables

Although it is often necessary to define local variables for use within the pulse sequence procedure (such as loop counters, calculated delays), only the global variables may be altered from within the RINMR data acquisition software.

Local variables should be declared at the beginning of the pulse sequence procedure following the VAR declaration, for example:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}

USES
{$I UNITS95.INC}

VAR
      PreAcqTime: REAL;

PreAcqTime:=(P90*US/2)-10*US;

PROCEDURE Sequence;

BEGIN

Duration(PreAcqTime*US,0);
```

```
END;

BEGIN
Run(Sequence);
END.
```

Programmers should note that constants used in expressions MUST have a time multiplier (either US, MS or S) associated with them.

2.8 Generating RF Pulses and Phase Cycling

RF pulses may be generated using the RF action function with the Duration procedure. The RF action function requires one argument that contains the phase list that the RF pulse will use. For example:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}

USES
{$I UNITS95.INC}

VAR

PROCEDURE Sequence;

BEGIN

ZeroTime;

Duration(1*US,RF(PH1));

Duration(RD*US,0);

END;

BEGIN
Run(Sequence);
END.
```

The ZeroTime command defines the time at which the sequence is started - the data display will define time zero at this point when the XYZERO option is selected from the view menu of the RINMR software (see the RINMR Manual for more information).

The first Duration procedure outputs an RF pulse of length 1 microsecond of phase PH1 and then the second Duration procedure waits for RD microseconds. The entire program will execute NS times (the variable NS is set via the RINMR acquisition software).

The phase list PH1 is specified in the RINMR parameter lists. PH1 may take the value 0, 1 , 2 or 3, corresponding to each of the quadrature phase values.

The programmer may increment the phase lists by using the Next procedure, which takes one argument, the phase list to be incremented. For example, to phase cycle the RF pulse in the program listed earlier:

```
PROGRAM MyPulseSequence;
```

```
{$I COMPILE.INC}

USES
{$I UNITS95.INC}

VAR

PROCEDURE Sequence;

BEGIN

ZeroTime;

Duration(1*US,RF(PH1));

Next(PH1);

Duration(RD*US,0);

END;

BEGIN
Run(Sequence);
END.
```

In the above program the phase list will be incremented every time a scan is executed. Programmers familiar with the MARAN MS-DOS software should note the different handling of phase lists using RINMR.

2.9 Data Acquisition

The MARAN internal hardware acquires data using a pair of analogue to digital converters (ADC's). The fastest acquisition rate is dependent on the configuration of the MARAN hardware and is 0.1 microseconds (10 MHz sampling rate) unless specified. Data are collected simultaneously in pairs. To acquire one pair of points the ADC action function is passed as an argument to the Duration procedure. The ADC action function also takes a phase list as a parameter, which specifies the acquisition phase in a similar manner to the RF pulse phase. In addition to the ADC function the receiver must be enabled by passing the REC action argument to the Duration procedure. The REC action argument must be set before the ADC action function to allow the receiver to stabilise.

To accumulate a series of points a Pascal loop must be used:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}

USES
{$I UNITS95.INC}

VAR
      n: LONGINT;

PROCEDURE Sequence;

BEGIN
ZeroTime;
```

```
Duration(10*US,REC);

For n:=1 to 512 DO
 Duration(2*US,ADC(PH2)+REC);

Next(PH2);

Duration(100*MS,0);

END;

BEGIN
Run(Sequence);
END.
```

The above program acquires 512 pairs of points 2 microseconds apart. Note that the receiver has to be enabled for 10 microseconds before the acquisition can take place and the local control variable n (a LONGINT) should be defined at the start of the program. If more than one scan is performed, the data is accumulated according to the phase list defined by PH2.

2.10 A Simple FID Sequence

Enough of the features of the MARAN pulse programming language have been described to allow a simple FID sequence to be written. The sequence is as follows:

```
Program FID;

USES{$UNITS95.INC}

VAR
      n: LONGINT;

PROCEDURE Sequence;

BEGIN

ZeroTime;

Duration(P90*US,RF(PH1));
Duration(Dead1*US,0);
Duration(Dead2*US,REC);

FOR n:=1 TO POINTS DO
 Duration(DWELL*US,ADC(PH2)+REC);

Next(PH1);
Next(PH2);

Duration(RD*US);

END;
```

```
BEGIN
Run(Sequence);
END.
```

2.11 Controlling Gradients

This section explains how to control the magnetic field gradient units supplied with the MARAN and MARAN Ultra systems. Programmers should note that not all MARAN systems are supplied with magnetic field gradients. All pulse programs that require gradients must specify the gradient unit under the USES section of the pulse program header:

```
Program GRADECHO;

USES
Grad,
{$UNITS95.INC}
```

Gradients are controlled using the gradient command. This command takes a single argument that specifies which gradient channels to activate:

```
Gradient(GX(G1)+GY(G1)+GZ(G2));
```

RINMR has two sets of parameters for controlling gradients. The first set G1-G9 allows the user to specify amplitude for individual gradient pulses. The second set GX, GY and GZ are channel scalars that multiply the output of each gradient channel by a specified amount.

The final gradient output of a particular channel is equal to the gradient pulse amplitude multiplied by the channel scalar. In the above example the amplitude of the output pulse will be G1*GX (and G1*GY, G1*GZ if specified). Note that all pulse amplitudes (G1-G9) are scaled by the current value of the channel scalar. Both the range of gradient variables G1 to G9 and GX, GY and GZ may take values between -32768 and +32767.

The final output strength of the gradient is given by the expression:

$$\text{Output} = \frac{\text{GN.G1}}{32768^2}\text{GMax}$$

where Gn is GX, GY or GZ and Gmax is the output gradient strength (G/cm or T/m).

Once the gradient command is issued, the gradients are activated until turned off by another gradient command:

```
Gradient(GX(OFF)+GY(OFF)+GZ(OFF));
```

To output a gradient pulse of 1 ms then following sequence of commands should be used:

```
Gradient(GX(G1)+GY(G1)+GZ(G1));
Duration(1*MS,0);
Gradient(GX(OFF)+GY(OFF)+GZ(OFF));
```

Programmers should note that due to the initialisation of the gradient hardware this series of statements produces a gradient pulse exactly 3μs longer than the 1ms specified in the duration command. In practice the shape and duration of the gradient pulse is dominated by the rise time of the gradient set/slew rate of the gradient amplifiers and the extra duration is insignificant. The extra time may be removed if required.

In the above example, the output on each of the three gradient channels is controlled using G1 from RINMR. As mentioned earlier, the scalars GX, GY, GZ must also be set within RINMR if gradients channels are to output correctly. For example if G1 is set to 100, GX is set to 32768, GY is set to 20000 and GZ is set to 0, only the X and Y gradient channels will output pulses. An alternative pulse sequence could specify a separate amplitude value for each gradient channel, using the following method:

```
Gradient(GX(G1)+GY(G2)+GZ(G3));
Duration (1*MS,0);
Gradient(GX(OFF)+GY(OFF)+GZ(OFF));
```

Now the X,Y and Z gradient channels can be controlled using the G1, G2 and G3 parameters. Using this technique GX, GY and GZ can be set to either 1 or 0 to toggle between on and off gradients.

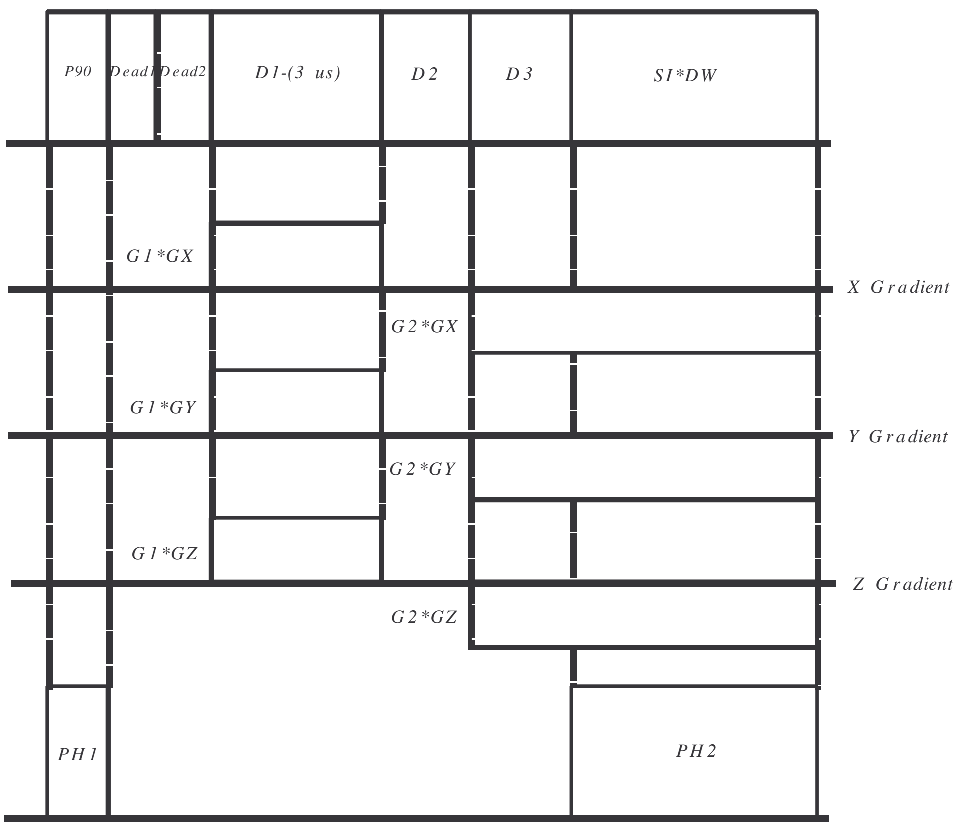
2.12 A Simple Gradient Echo Sequence

We are now in a position to define a simple gradient echo sequence for performing a one-dimensional profile.

```
PROGRAM GRADECHO;

USES
Grad,
{$UNITS95.INC}

VAR
      n: LONGINT;

PROCEDURE Sequence;

BEGIN

ZeroTime;

Duration(P90*US,RF(PH1))
Duration(Dead1*US,0)
Duration(Dead2*US,0)
Gradient(GX(G1)+GY(G1)+GZ(G1));
Duration(D1*US,0);
Gradient(GX(OFF)+GY(OFF)+GZ(OFF));
Duration(D2*US,0);
Gradient(GX(G2)+GY(G2)+GZ(G2));
Duration(D3*US,REC);

FOR n:=1 to SI DO
 Duration(DW*US,ADC(PH2)+REC);

Gradient(GX(OFF)+GY(OFF)+GZ(OFF);

Next(PH1);
Next(PH2);

Duration(RD*US,0);

END;
```

```
BEGIN
Run(Sequence);
END.
```

A timing diagram for the sequence may be found below. Note that the second gradient pulse should be negative for correct re-phasing of the gradient echo, which is achieved by specifying negative values for G1-G9 and GX, GY and GZ.



Timing Diagram for the GRADECHO Sequence

Chapter 3 Pulse Programming Using MARAN Ultra Hardware

3.1 Introduction

The set of procedures and functions, which define the events, and timing of an NMR experiment are known as a pulse sequence or pulse program. A typical pulse sequence specifies a number of RF pulses, phase programs, timing calculations and acquisition delays. MARAN Ultra pulse sequences are written in Turbo Pascal and are created and edited using the Turbo Pascal editor. Once a pulse sequence has been written it must be compiled into an executable before it can be used with the RINMR data acquisition software.

3.2 Specifying the Default Editor and Compiling the Pulse Sequence

The default pulse sequence editor must be specified in the RINMR software before pulse sequence editing and compilation can take place (the RINMR User Manual contains information on how to do this). The RINMR Developers Kit is supplied with Borland Turbo Pascal 7.0 for pulse sequence compilation. Users should ensure that the correct path (usually c:\tp\bin\turbo.exe) and correct default pulse sequence directory (usually c:\program files\resonance\RINMR\seq\source) are specified under the Tools menu of the RINMR software before attempting to write and compile pulse programs.

3.3 Pulse Program Skeleton

The pulse sequence must be defined as a Pascal procedure called Sequence. The following listing can be used as a skeleton and is stored on disk in the c:\program files\resonance\RINMR\seq\source directory:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}
USES
{$I UNITS.INC}
VAR
{Define local variables here}
PROCEDURE Sequence;
BEGIN
 {Enter pulse sequence here}
END;
BEGIN
 Run(Sequence);
END.
```

3.4 Global Pulse Sequence Variables

A set of global, pre-defined variables exists for use in the pulse sequences. All the pre-defined variables may be accessed using the RINMR data acquisition software in acquisition mode.

RINMR Parameter Name	Intended Use
P90	90 degree pulse length
P180	180 degree pulse length
SI	Size of acquisition buffer
TAU	Delay
DEAD1	Probe dead time
DEAD2	Filter settle time
RD	Experiment recycle delay
DW	Sample frequency
NS	Number of scans
PH1-PH5	Phase programs
D1-D12	Delays

P1-P5	Pulses
C1-C12	Counters (integer)
FP1-FP6	Counters (floating point)
Nech	Number of echoes
G1-G9	Gradient amplitudes
GX	X gradient global scalar
GY	Y gradient global scalar
GZ	Z gradient global scalar
IG1-IG9	Incrementing Gradients
SH1-SH5	Shaped RF Event Identifiers
RFA0-RFA5	RF Event scalars (RF Channel 1)
RF2A0-RF2A5	RF Event scalars (RF Channel 2)
GSH1-GSH5	Shaped Gradient Event Identifiers

3.5 OIMBL Pulse Programming Conventions

Pulse programs written by OIMBL follow a convention to facilitate the operation of pulse sequences by users. It is recommended that pulse sequences written by programmers adhere to this convention.

Phase Programs

The first event (which is nearly always an RF pulse) is assigned phase list PH1.
The ADC phase list is defined as PH2.
The second event phase list is defined as PH3.
The third event phase list is defined as PH4 etc.

Delays

Tau is used as the time delay between consecutive RF pulse centres (for example in the CPMG pulse sequence).

In other pulse sequences where a duration event is specified from the end of one RF pulse to the beginning of another, a Dn variable is used for the duration (for example the SOLID pulse sequence, CPMAS sequence).

RF Pulses

P90 is used where a 90-degree pulse is required.
P180 is used where a 180-degree pulse is required.

Gradients

The first gradient event is assigned amplitude G1.
The second gradient event is assigned amplitude G2 etc.

ZeroTime

Zerotime is defined as the start of the first RF event.

It should be noted that certain pulse sequences, such as DROPLET, do not adhere to this standard in order to allow easier setting up procedures.

3.6 Information Files

In order to facilitate the use of pulse sequences, programmers may define an information file that describes the parameters used in the pulse sequence and their function.

The information file should be a ASCII text file (created with Windows Notepad or a similar text editor) comprised of a list of the parameters used in the sequence and their description. For example, the information for the pulse sequence such as FID might be as follows:

```
P90         90 Degree Pulse (us)
Dead1       Probe Dead Time (us)
Dead2       Receiver Dead Time (us)
SF          Spectrometer Frequency (MHz)
O1          Offset from SF (Hz)
FW          Filter Width (Hz)
DW          Dwell Time (us)
SI          Size of Acquisition Buffer (points)
NS          Number of Scans
RG          Receiver Gain (%)
RD          Relaxation Delay (us)
PH1         90 Degree Pulse Phase List (rec: 0213)
PH2         Receiver Phase List (rec: 0213)
DS          Dummy Scans
RFA0        RF Amplitude (%)
```

The information file should be placed in the C:\Program Files\RINMR\Seq\Bin directory and given the extension *.info. The information contained in the file will be displayed in RINMR when the user views the acquisition parameters. Note that if no information file is created, RINMR will use the default information file, default.info, when the acquisition parameters are displayed.

3.7 The Duration Procedure

Pulse programs are comprised of a series of Duration procedures. The Duration procedure takes two arguments. The first argument is a variable of type LONGINT or a constant that defines the length in time of the Duration procedure. The second argument (action argument/function) tells the hardware what actions to perform during the Duration procedure. For example:

```
Duration(10*US,0)
```

This command instructs the pulse sequencer to do nothing for 10 microseconds. In addition to the constant, a time multiplier (either US, MS or S) must be specified.

The pre-defined variables may specified as arguments to the Duration procedure, for example:

```
Duration(D1,0)
```

This command instructs the pulse sequencer to do nothing for D1 microseconds. D1 may be altered interactively from within the RINMR data acquisition software.

The different types of action arguments/functions that may be passed to the Duration command are listed in the following table:

Argument	Description	Example
0	Do nothing	Duration(D1,0)
RF(PHn)	Output on the first RF channel, use phase program PHn	Duration(P90,RF(PH1))
RF2(PHn)	Output on the second RF channel, use phase program PHn	Duration(P90,RF2(PH4))
FRF(PHn)	Output a fast RF pulse on the first RF channel, use phase program PHn	Duration(P90,FRF(PH3))
FRF2(PHn)	Output a fast RF pulse on the second RF	Duration(P90,FRF2(PH2))

	channel, use phase program PHn	
ADC(PHn)	Acquire one pair of points (simultaneously), use phase program PHn	Duration(DW,ADC(PH2)
REC	Enable the receiver	Duration(DEAD2,REC)
TXEnable1	Enable the first RF channel	Duration(8*US,TXEnable1)
TXEnable2	Enable the second RF channel	Duration(8*US,TXEnable2)
CWn	Output an RF pulse on the CW channel, phase n.	Duration(P90,CW3)

More than one action argument/function may be passed using the addition operator, for example:

Duration(D1,REC+ADC)

More information on the use of specific action arguments is contained in sections 3.9-3.13.

3.8 Local Pulse Sequence Variables

Although it is often necessary to define local variables for use within the pulse sequence procedure (such as loop counters, calculated delays), only the global variables may be altered from within the RINMR data acquisition software.

Local variables should be declared at the beginning of the pulse sequence procedure following the VAR declaration, for example:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}

USES
{$I UNITS.INC}

VAR
     PreAcqTime: LONGINT

PreAcqTime:=(P90/2)-10*US;

PROCEDURE Sequence;

BEGIN
 Duration(PreAcqTime,0);
END;

BEGIN
 Run(Sequence);
END.
```

Programmers should note that constants used in expressions MUST have a time multiplier (either US, MS or S) associated with them.

3.9 Generating RF Pulses and Phase Cycling

RF pulses may be generated using the RF action function with the Duration procedure. The RF action function requires one argument that contains the phase list that the RF pulse will use. For example:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}
```

```
USES
{$I UNITS.INC}

VAR

PROCEDURE Sequence;

BEGIN
      Duration(20*US,TXEnable1);
      Zerotime;
      Duration(1*US,RF(PH1)+TXEnable1);
      Duration(RD,0);
END;

BEGIN
Run(Sequence);
END.
```

The first Duration procedure takes the TXEnable1 action argument that enables the transmitter for 20µs before the RF pulse is applied. All MARAN Ultra RF transmitters must be enabled for 20µs before RF events can occur (this is sometimes known as turning off the blanking). Programmers writing pulse sequences for use with other RF amplifiers should refer to the amplifier documentation supplied with the RF amplifier for information on enable times and blanking control.

The `zerotime` command defines the start time of the sequence for RINMR - the data display will define time zero at this point when the XYZERO option is selected from the view menu of the RINMR software (see the RINMR User Manual for more information).

The third Duration procedure outputs an RF pulse of length 1µs of phase PH1 and then waits for RD microseconds. The entire program will execute NS times (the variable NS is set via the RINMR software).

The phase program PH1 is assigned in the RINMR data acquisition software. PH1 may take the value 0, 1, 2 or 3, corresponding to each of the quadrature phase values.

The final Duration procedure instructs the pulse program to do nothing for RD microseconds.

The programmer may increment the RF pulse phase by using the Next procedure, which takes one argument, the phase list to be incremented. For example, to phase cycle the RF pulse in the program listed earlier:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}

USES
{$I UNITS.INC}

VAR

PROCEDURE Sequence;

BEGIN
      Duration(20*US,TXEnable1);
      Duration(2*US,RF(PH1));
      Next(PH1);
      Duration(RD,0);

END;

BEGIN
Run(Sequence);
END.
```

In the above listing the phase of the RF pulse will be changed according to the PH1 phase list every time a scan is executed. Programmers should note that two RF channels exist for spectroscopy sequences, RF and RF2. The receiver is locked to the first RF channel (RF).

3.10 Data Acquisition

Data is acquired by the spectrometer using a pair of analogue to digital converters (ADC's). The fastest acquisition rate is dependent on the configuration of the hardware and is 0.1μs (10 MHz sampling rate) unless otherwise indicated. Data is collected simultaneously in pairs. To acquire one pair of points the ADC action function is passed as an argument to the Duration procedure. The action function ADC also takes a phase list as a parameter, which specifies the acquisition phase in a similar manner to the RF pulse phase. In addition to the ADC action function, the REC action argument is used to enable the receiver in the Duration procedure. The REC action argument must be specified before the ADC action function is used to allow the receiver to stabilise.

To accumulate a series of points a Pascal loop must be used:

```
PROGRAM MyPulseSequence;
{$I COMPILE.INC}

USES
{$I UNITS.INC}

VAR
      n: LONGINT;

PROCEDURE Sequence;

BEGIN
      Duration(10*US,REC);

      For n:=1 to 512 DO
       Duration(2*US,ADC(PH2)+REC);
      Next(PH2);

      Duration(100*MS,0);

END;

BEGIN
Run(Sequence);
END.
```

The above program acquires 512 pairs of points spaced 2μs apart. Note that the receiver has to be enabled for 10 microseconds before the acquisition can take place and the local control variable n (a LONGINT) for the acquisition loop must be defined at the start of the program. If more than one scan is performed, the data are accumulated according to the phase list defined by PH2.

3.11 A Simple FID Sequence

Enough of the features of the RINMR pulse programming language have been described to allow a simple FID sequence to be written. The sequence is as follows:

```
Program FID;

USES{$UNITS.INC}

VAR
      n: LONGINT;

PROCEDURE Sequence;

BEGIN
      Duration(20*us,TXEnable1);

      ZeroTime;

      Duration(P90,RF(PH1)+TXEnable1);
      Duration(Dead1,0);
      Duration(Dead2,REC);
      FOR n:=1 TO SI DO
       Duration(DW,ADC(PH2)+REC);

      Next(PH1);
      Next(PH2);

      Duration(RD,0);

END;
```

3.12 Fast RF Pulse Generation

The pulse programming language has a facility for generating fast RF pulses. Fast RF pulses are generated using the FRF action function:

```
Duration(FRF(PH1));
```

Fast RF pulses have two properties:

- They can be used to switch phase instantaneously (useful for spectroscopy applications/spin lock pulse generation).
- They do not use so many events in the pulse generator (useful for sequences in which large numbers of events are required, e.g. CPMG sequences).

3.13 Multi Channel RF Pulse Sequences

The MARAN Ultra Spectroscopy Hardware has a number of sophisticated features for high-resolution spectroscopy applications. The following listing gives an example of a multi channel pulse sequence used for cross polarisation:

```
PROGRAM CPMAS;
```

```
{$I COMPILE.INC }

USES
{$I UNITS.INC }

VAR
      n: LONGINT;

PROCEDURE Sequence;

BEGIN
Duration(20*us,TXEnable1+TXEnable2);

ZeroTime;
Duration(P90,FRF2(PH1)+TXEnable1+TXEnable2);
Duration(D5,FRF(PH2)+TXEnable1+CW3);
Duration(Dead1,CW3);
Duration(Dead2,REC+CW3);

FOR n:=1 TO SI DO
 Duration(DW,ADC(PH3)+REC+CW3);

Next(PH1);
Next(PH2);
Next(PH3);

Duration(RD,0);

END;

BEGIN
Run(Sequence);
END.
```

The sequence for cross polarisation requires two RF channels, in this specific example one for the ^{13}C NMR excitation and one for the ^{1}H excitation. A Pulse Sequence Timing diagram can be seen in Figure 3.1.

The first Duration procedure enables both RF amplifiers. The second Duration procedure outputs a fast RF pulse on the second (^{1}H) channel (FRF2). Note that both the RF amplifiers are enabled during the execution of this procedure and the RF argument is replaced by the FRF2 to enable fast phase switching.

The third Duration procedure activates the first (^{13}C) channel for D5 microseconds and locks the second RF channel on with the CW3 action argument (for continuous wave operation, 3rd phase).
The fourth and fifth Duration procedures wait for the duration DEAD1 and DEAD2 microseconds respectively. The receiver is enabled during DEAD2. Programmers should note that the receiver channel is locked to the frequency of the first RF channel (FRF) and should also note that the second RF channel (FRF2) is on throughout the acquisition and dead time period.

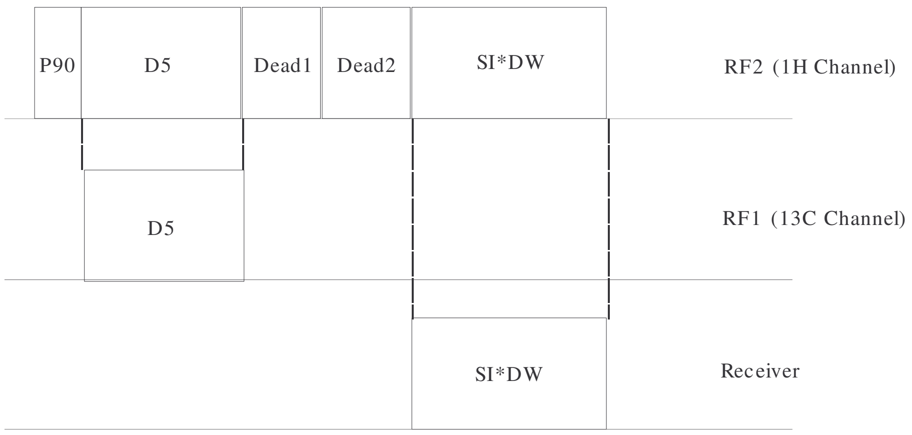


Figure 3.1 Pulse Sequence Timing Diagram for CP sequence.

3.14 Controlling Gradients

This section explains how to control the magnetic field gradient units. Programmers should note that not all MARAN Ultra Spectroscopy systems are supplied with magnetic field gradients.

All pulse programs that require gradients must specify the gradient unit under the USES section of the pulse program header:

```
Program GRADECHO;
{$COMPILE.INC}

USES
Grad,
{UNITS.INC}
```

Gradients are controlled using the Gradient procedure. This procedure takes a single argument which specifies which gradient channels to activate:

```
Gradient(GX(G1)+GY(G1)+GZ(G2))
```

RINMR has two sets of parameters for controlling gradients. The first set G1-G9 allows the user to specify amplitudes for individual gradient pulses. The second set, GX, GY and GZ, are global channel scalars that multiply the output of all gradient pulses on each gradient channel by a specified amount.

The final amplitude output of a particular gradient channel is equal to the gradient pulse amplitude multiplied by the channel scalar. In the above example the amplitude of the output pulse will be G1.GX (and G1.GY, G1.GZ if specified). Note that all pulse amplitudes (G1-G9) are scaled by the current value of the channel scalar. Variables G1-G9 may take values between -32767 and 32768 and variables GX, GY and GZ may take values between 0 and +32767.

Once the Gradient procedure is called, the gradients are activated until turned off by another Gradient procedure:

```
Gradient(GX(OFF)+GY(OFF)+GZ(OFF))
```

To output a gradient pulse of 1 ms the following sequence of commands should be used:

```
Gradient(GX(G1)+GY(G1)+GZ(G1));
Duration(1*ms,0);
Gradient(GX(OFF)+GY(OFF)+GZ(OFF));
```

Programmers should note that due to the initialisation of the gradient hardware this series of statements produces a gradient pulse exactly 3µs longer than the 1ms specified in the Duration procedure. In practice the shape and duration of the gradient pulse is dominated by the rise time of the gradient set/slew rate of the gradient amplifiers and the extra duration is insignificant. The extra time may be removed if required.

In the above example, the output on each of the three gradient channels is controlled using G1 from RINMR. As mentioned earlier, the scalars GX, GY, GZ must also be set within RINMR if gradients channels are to output correctly. For example if G1 is set to 100, GX is set to 32768, GY is set to 20000 and GZ is set to 0, only the X and Y gradient channels will output pulses.

An alternative pulse sequence could specify a separate amplitude value for each gradient channel, using the following method:

```
Gradient(GX(G1)+GY(G2)+GZ(G3));
Duration (1*MS,0);
Gradient(GX(OFF)+GY(OFF)+GZ(OFF));
```

Now the X,Y and Z gradient channels can be controlled using the G1, G2 and G3 parameters. Using this technique GX, GY and GZ can be set to either 32767 or 0 to toggle between on and off gradients.

3.15 A Simple Gradient Echo Sequence

We are now in a position to define a simple gradient echo sequence for performing a one-dimensional profile:

```
PROGRAM GRADECHO;
{$I COMPILE.INC}

USES
Grad,
{$UNITS.INC}

VAR
      n: LONGINT;

PROCEDURE Sequence;

BEGIN
Zerotime;
Duration(P90,RF(PH1));
Duration(Dead1,0);
Duration(Dead2,0);
Gradient(GX(G1)+GY(G1)+GZ(G1));
Duration(D1,0);
Gradient(GX(OFF)+GY(OFF)+GZ(OFF));
Duration(D2,0);
Gradient(GX(G2)+GY(G2)+GZ(G2));
Duration(D3,REC);

FOR n:=1 to SI DO
```

```
      Duration(DW,ADC(PH2)+REC);

Gradient(GX(OFF)+GY(OFF)+GZ(OFF);

Next(PH1);
Next(PH2);

Duration(RD,0);

END;

BEGIN
Run(Sequence);
END.
```

A timing diagram for the sequence may be found below. Note that the second gradient pulse should be negative for correct rephasing of the gradient echo, which is achieved by specifying negative values for the range of gradient variables G1 to G9 and GX, GY and GZ (Figure 3.2).

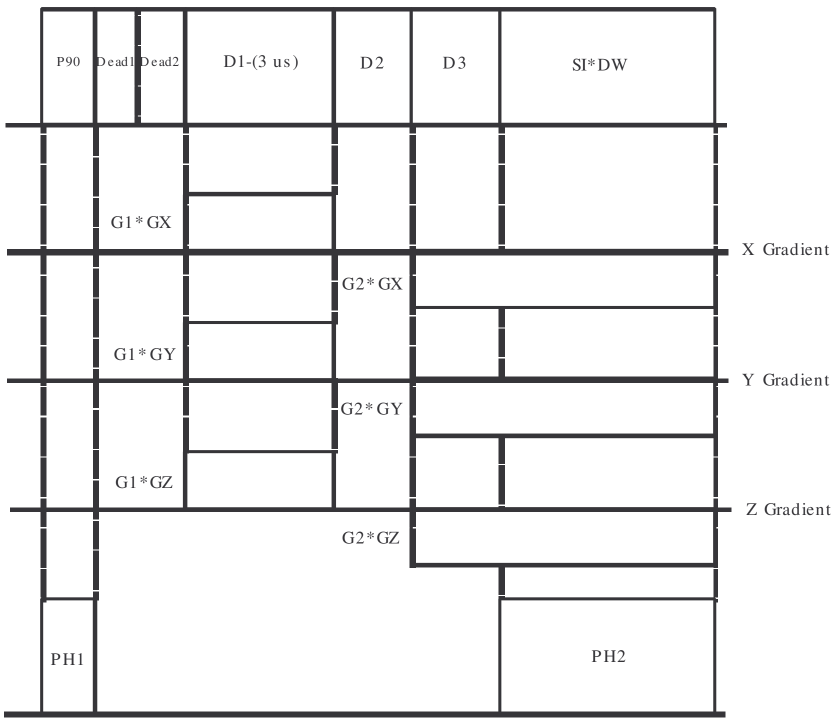


Figure 3.2 Timing Diagram for the GRADECHO Sequence

3.16 Incrementing Gradients

Incrementing gradients are necessary in order to conduct imaging sequences. The I Prefix is used to specify an incrementing gradient in the gradient procedure:

```
Gradient(IGX(G1));
```

G1 holds the current value of the incrementing gradient (note that this may not be accessed by the programmer). The number of times the gradient is to increment is held by the variable IG1 that is specified by the user within the RINMR data acquisition software, along with GX that should be assigned the maximum value of the incrementing gradient. From the parameters IG1 and GX, the software automatically calculates the necessary gradient values for each phase encode step.

Thus the value of the gradient scalar is automatically divided into IG1 phase encode steps such that the IG1th/2 value will be zero. This removes the need for the user to specify a start gradient value and gradient increment. The user simply specifies the maximum value of the phase encode gradient through the gradient scalar and the number of phase encode steps required.

Gradients may be incremented using the NextGrad procedure: `NextGrad(IG1);` that increment the gradient IG1 to its next value.

Sequences with incrementing gradients also require a loop back command to inform the hardware to return to the beginning of the pulse sequence and re-execute. This is performed using the LOOP procedure that uses the following syntax: `Loop(IG1);` that instructs the hardware to execute the pulse sequence IG1 times.

The following extract from a single point imaging pulse sequence shows how to use incrementing gradients as part of a pulse program:

```
Duration(10*US,0);
Gradient(IGX(G1)+TxEnable);
Duration(D1,TxEnable);
Duration(P90,RF(PH1)+TxEnable);
Duration(D2,0);
Duration(DW,ADC(PH2)+REC);
Gradient(IGX(OFF));
Duration(RD,0);
Next(PH1);
Next(PH2);
Loop(NS)
NextGrad(IG1);
Loop(IG1);
END;
```

The sequence starts by turning on the incrementing gradient IG1 and allowing the X gradient to stabilise for a period D1. The gradient is applied continuously during the 90^o pulse and afterwards for a period D2, when a data point is acquired.

This process is repeated for NS scans for each value of incrementing gradient using the Loop(NS) procedure. Note that the Loop(NS) procedure is implicit in pulse sequences which do not contain incrementing gradients (such as the GRADECHO or FID pulse sequences). In pulse sequences that contain incrementing gradients the hardware requires information on when to loop for scans and when to loop for incrementing gradients. In pulse sequences with incrementing gradients the hardware has to be told explicitly when to loop for gradients and when to loop for scans.
Users should note that the above program is an example listing only. Additional procedures concerning data bin allocation are necessary in order for the program to function correctly.

3.17 Shaped RF Events

Shaped RF events may only be produced by MARAN Ultra systems equipped with special shaped RF event hardware.

All pulse sequences using shaped RF events must specify the RFSHAPE unit under the USES section of the program header:

```
PROGRAM Shaped90;
{$I COMPILE.INC}

USES
RFShape,
{$I UNITS.INC}
```

Shaped RF events are generated in the pulse program using the ShapedRF procedure, which takes four arguments:

```
ShapedRF(Time, Phase, Identifier, Amplitude, Enable);

e.g. ShapedRF(P1, PH1, SH1, RFA1, TxEnable1).
```

The arguments are as follows:

- *Time*: The time for which the shaped RF event is generated. Note that according to the RI pulse programming convention the variables P1-P5 are used for shaped RF events. P90 and P180 are used for hard RF pulses.
- *Phase*: The phase program for the shaped RF pulse.
- *Identifier*: The identifier (which associates a shape file to the shaped RF event) to use.
- *Amplitude*: An overall amplitude scalar for the shaped RF event. Note that RFA0 always corresponds to the amplitude of hard RF events generated with the RF action function. According to RI pulse programming convention, the first shaped RF event should be designated RFA1.
- *Enable*: Enable the RF amplifier.

To produce a shaped RF event a shape file containing a description of the shape must be generated.

- The shape file should be an ASCII file containing two columns of numbers, the first column contains the amplitude of the shaped RF event and the second column contains the phase of the RF event.
- The numbers should and range from 0 to 1 for amplitude and 0 to 2π for phase.
- Once the file has been created it must be stored in the c:\Program Files\Resonance\RINMR\shape directory.
- Note that if the shape file is not placed in the correct directory it will not load.
- The programmer must specify an identifier parameter (SH1-SH5) for each shaped RF event, which allows the user to associate a shape file with the event.
- To identify a file with a particular shaped RF event within RINMR, type in the command line: `SHn filename` (SH1 gauss256, in the above example).
- In addition to the shape file, the overall amplitude of the shape may be scaled via the amplitude argument. The pre-defined global variables RFAn should be used for this purpose. The phase program is handled in the usual manner.
- Finally, the Enable argument allows the relevant RF amplifier to be enabled (i.e. either TXEnable1 or TXEnable2).

The following program provides an example of shaped RF pulses:

```
PROGRAM Shaped90;
{$I COMPILE.INC }

USES
      RFShape,
{$I UNITS.INC }
```

```
VAR
      n,i: LONGINT;

PROCEDURE Sequence;

BEGIN

ZeroTime;

Duration(10*us,0);
Duration(D1,TXEnable1);
ShapedRF(P1,PH1,SH1,RFA1,TXEnable1);
Duration(Dead1,0);
Duration(Dead2,0);
FOR n:=1 TO SI DO
 Duration(DW,ADC(PH2)+REC);
Duration(RD,0)
Next(PH1);
Next(PH2);
END;

BEGIN
Run(Sequence);
END.
```

3.18 Shaped Gradient Events

Shaped gradient events may be generated only by MARAN Ultra systems equipped with special shaped gradient event hardware. Programmers must specify the Grad unit in any pulse programs that use shaped gradient events. Shaped gradient events may be generated using the ShapeGrad function, which takes the four arguments:

```
Time:=ShapedGrad(Timebase, Identifier, Channel, Lines);
```

The arguments are as follows:

- *Timebase*: The duration per point in the shape file (example D1, 100*US).
- *Identifier*: The identifier that associates a shape file with the shaped gradient event (example GSH1).
- *Channel*: The gradient channel that will output the shaped event. Multiple channels may be specified using the + operator.
- *Lines*: Any other control lines that are to be activated during the output of the event.

The ShapedGrad function returns the time for which the shaped gradient event occurs (number of points in gradient shape file multiplied by timebase). Once the ShapeGrad function is called, it takes 18µs until the shaped gradient is started.

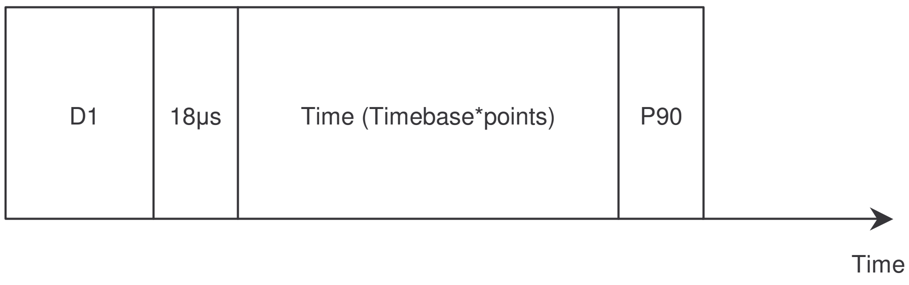
To produce a shaped gradient event a shape file containing a description of the shape must be generated. The shape file should be an ASCII file containing a single column of numbers that describes the amplitude of the gradient event at a particular time. The numbers will be scaled automatically to use the full dynamic range of the hardware. Once the file has been created it must be stored in the c:\Program Files\Resonance\RINMR\shape directory.

To associate a file with a particular shaped gradient event within RINMR, type in the command line: `GSHn filename`, for example “GSH1 sinusoid”.

Note the final output amplitude of the shaped gradient event will be proportional to GN (where n=X,Y,Z) * GX (where x=1-5) * value in Shape file.

The following programs provide examples of the use of shaped gradient pulses:

```
Duration(D1,0);
Time:=ShapedGrad(100*us,GSH1,GX(G1),0);
Duration(Time,0);
Duration(P90,RF(PH1));
```



During the duration Time the shape described by GSH1 will be played out on the X channel.

```
Time=ShapedGrad(D1,GSH2,IGY(G1),0)
Duration(123*us,0);
Duration(1*us,ADC(PH2));
Duration(Time-((123+1)*us),0);
Duration(1*us,ADC(PH2));
..
..
NextGrad(G1);
```

The above program will play out the shape identified with GSH2 on the Y channel. Each point will last D1 microseconds. The amplitude of the gradient shape will change with each scan, from negative to positive, the number of incrementing steps is defined in RINMR by IG1.

123 microseconds after the shaped gradient has started, a single data point is acquired. Immediately after the shaped gradient event has finished a second point is acquired.

```
Time:=ShapedGrad(D1,GSH1,GX(G1)+GY(G2)+GZ(G3),0);
```

In the above case the same shape is played out on all three gradient channels simultaneously. The amplitude of each gradient (X,Y,Z) may be set differently.

3.19 Multi-Dimensional Data Acquisition and the Loop Command

The RINMR pulse programming language has a flexible multi-dimensional data acquisition capability to cope with the demands of both imaging and two-dimensional spectroscopy sequences.

To set up a multi-dimensional data acquisition buffer, the MultiAcquire procedure should be used. This takes the following syntax:

```
MultiAcquire(DIM1,DIM2,DIM3,DIM4,ACQUIRE);
```

where the DIM1, DIM2, DIM3, DIM4 are the dimensions of the data buffer and acquire is the name of a control procedure to be called after each DIM1 points are acquired.

The acquisition buffer should be set up using a separate procedure listed in the pulse program before the pulse sequence. For example:

```
Procedure SetAcquire;

BEGIN
MultiAcquire(SI,64,1,1,Acquire);
END.
```

In addition, any extra control variables that are required may be initialised in this procedure.

The above procedure initialises a multi dimensional acquisition buffer of dimensions SI * 64 * 1 * 1.

Every time DIM1 points are placed into the buffer the procedure Acquire (defined in a separate part of the pulse program) is called to allow the user to manipulate where in the data buffer the next set of data will be placed. The acquisition buffer is divided into DIM2*DIM3*DIM4 data bins, each of which holds DIM1 points. The data bin into which the next DIM1 points are placed is specified using the DataBin command that takes the syntax:

```
DataBin(Bin_number);
```

Where Bin_number is the number of the data bin where the next DIM1 points will be placed.

The following FID sequence illustrates the use of multidimensional acquisition buffers:

```
PROCEDURE Acquire;

BEGIN
IF ANS=NS THEN
      BEGIN
      ANS:=0;
      INC(Bin);
      END;
DataBin(Bin);
INC(ANS);
END;

Procedure SetAcquire;
BEGIN
MultiAcquire(SI,64,1,1,Acquire);
END.
Procedure Sequence;

BEGIN

ZeroTime;
      Duration(10*us,TXEnable1);
      Duration(P90,RFA(PH1)+TXEnable1);
      Duration(Dead1,0);
      Duration(Dead2,0);
      FOR n:=1 TO SI DO
       Duration(DW,ADC(PH2)+REC);
      Duration(RD,0);
      Next(PH1);
      Next(PH2);
      LOOP(NS);
      LOOP(64);
END;
```

The above sequence performs NS*64 scans. Before each SI points are acquired, the Acquire procedure is called. This procedure checks whether NS scans have been performed (if NS scans have not been performed we are in the middle of a phase cycle and the data should be added to the current data bin according to the phase lists); in the procedure, a temporary variable ANS is compared with NS. If ANS is not yet equal to NS, ANS is incremented and the data bin is left unchanged.

If NS scans have been performed, the data bin is incremented using the DataBin command (the value of the current data bin is held in the global variable bin) and ANS is reset.

The two loop procedures instruct the hardware to return to the beginning of the pulse sequence at the end of each series of scans. The first LOOP procedure instructs the hardware to return NS times, the second 64 times, resulting in NS scans being performed for each of the 64 data bins. Note that for a multi-dimensional acquisition LOOP(NS) must be stated explicitly. For a 1 dimensional acquisition LOOP(NS) is implicit within the pulse sequence and does not need to be stated.

3.20 Varying the Pulse Amplitude within a Sequence for Spin Locking

The hard RF pulse amplitude may be varied within a pulse sequence using the SetRFPower procedure. This procedure takes a single argument, the name of a procedure to be called to set the power level of all subsequent hard RF pulses. The called procedure sets the RF power via the LoadRFPower procedure, which requires two arguments, a dummy variable that contains an internal reference for the pulse sequence compiler and a variable that specifies the amplitude of the hard RF pulses (usually RFAx where x=1 to 5). The syntax is as follows:

```
PROCEDURE Power90(Loc:Longint);
BEGIN
LoadRFPower(Loc,RFA0); { Define hard RF pulse power level }
END;
SetRFPower(Power90); {Set hard RF power using Power90 procedure}
Duration(P90,RF(PH1)+TXEnable1); { Spin locking pulse, phase PH1}
```

All RF pulses following the SetRFPower statement will be output using the specified RF power (in the above case RFA0). A typical spin lock pulse sequence can be written as follows:

```
PROGRAM T1rho;

{$I COMPILE.INC }

USES
{$I UNITS.INC }

VAR
n: LONGINT;

PROCEDURE Power90(Loc:Longint);
```

```
BEGIN
LoadRFPower(Loc,RFA0);            { Define 90 degree power level}
END;

PROCEDURE PowerLock(Loc:Longint);
BEGIN
LoadRFPower(Loc,RFA1);            {Define power level of spin locking
pulse}
END;

PROCEDURE Sequence;

BEGIN
Duration(20*us,TXEnable1);        {Enable CH1 RF}
ZeroTime;                         {Define Start Time For RINMR}
SetRFPower(Power90);              {Set RF power using RFA0}
Duration(P90,FRF(PH1)+TXEnable1);{Fast 90 Degree Pulse, Phase PH1}
SetRFPower(PowerLock);            {Set RF power using RFA1}
Duration(D1,FRF(PH3)+TXEnable1); {Spin Locking Pulse (fast), Phase PH3}
Duration(Dead1,0);                {Wait For Probe Ring Down}
Duration(Dead2,REC);              {Filter Settle, Enable Receiver}
FOR n:=1 TO SI DO
 Duration(DW,ADC(PH2)+REC);       {Acquire Echo, Phase PH2}

Next(PH1);                        {Increment Phase Lists}
Next(PH2);
Next(PH3);

Duration(RD,0); {Wait For Relaxation Delay}

END;

BEGIN
Run(Sequence);
END.
```

Note the use of the FRF function to generate the spin lock pulse. This ensures that the transition between the initial 90º pulse and the spin lock pulse is instantaneous.

Chapter 4 Library of Pulse Sequences and automated Scripts

The RINMR Pulse Sequence Manual contains information on the pulse sequences supplied with the RINMR software for both MARAN Ultra and MARAN hardware. Note that the pulse sequences may vary between instrument configurations.

4.1 Start Up Parameters

When RINMR pulse sequences are executed for the first time they use a system default parameter set as a starting point. At this point the phase lists for each pulse sequence will not be set correctly. Phase lists must be set before pulse sequences such as HAHN, INVREC, CPMG and SOLID will operate correctly. Recommended phase lists for each pulse sequence are provided with this document and can also be found under the RINMR acquisition parameters menu. Once the phase lists are entered for each pulse sequence they will not need to be altered. Note that pulse sequence phase lists are not configured automatically on installation and must be set up before NMR experiments are performed.

4.2 Pulse Sequences & Scripts

Note that some pulse sequences, such as PROFILE and CPMAS require additional hardware in order to run correctly, and that two types of hardware exist, MARAN hardware and MARAN Ultra hardware. Not all pulse sequences are available for MARAN hardware.

Pulse Sequences described in this manual are as follows:

Sequence Name	Hardware Types	Special Hardware Required
FID	MARAN / Ultra	None
CPMG	MARAN / Ultra	None
CPMGF	Ultra	None
CPMGGRAD	Ultra	Gradients
FIDRF2	Ultra	2nd RF Channel
FIDHCW	Ultra	2nd RF Channel and decouple transmitter
CPMAS	Ultra	2nd RF Channel and decouple transmitter
INVREC	MARAN / Ultra	None
HAHN	MARAN / Ultra	None
FID_HAHN	MARAN / Ultra	None
SOLID	MARAN / Ultra	None
PROFILE	MARAN / Ultra	Gradients
WOBBLE	Ultra	Gradients
DIFF	MARAN / Ultra	Gradients
DIFFA	Ultra	Gradients
DROPLET	Ultra	Gradients
MULTIFID	Ultra	Gradients and Shaped Pulse Generator
TRAIN90	Ultra	None
TRAIN180	Ultra	None
FID_ONE	Ultra	None
SETT1RHO	Ultra	T1 Rho Option
T1RHO	Ultra	T1 Rho Option

Script Name	Hardware Types	Special Hardware Required
GETDIFF	MARAN / Ultra	Gradients
GETDIFFA	MARAN / Ultra	Gradients
T1	MARAN / Ultra	None
GETDROP	Ultra	Gradients/Droplet Size Analysis
GEGAUSS	MARAN / Ultra	Gradients
LISTGEN	MARAN / Ultra	None
PROBECHANGE	MARAN / Ultra	None
GRADCALIB	MARAN / Ultra	Gradients
GETT1RHO	Ultra	T1 Rho Option
TD	MARAN / Ultra	None
IMAGESET	Ultra	Imaging
SL90XYZ	Ultra	Imaging
IMAG90XY	Ultra	Imaging
IMAG90XZ	Ultra	Imaging
T190XY	Ultra	Imaging
MULSLXY	Ultra	Imaging
MULSLXZ	Ultra	Imaging
MULECHXY	Ultra	Imaging
MULECHXZ	Ultra	Imaging

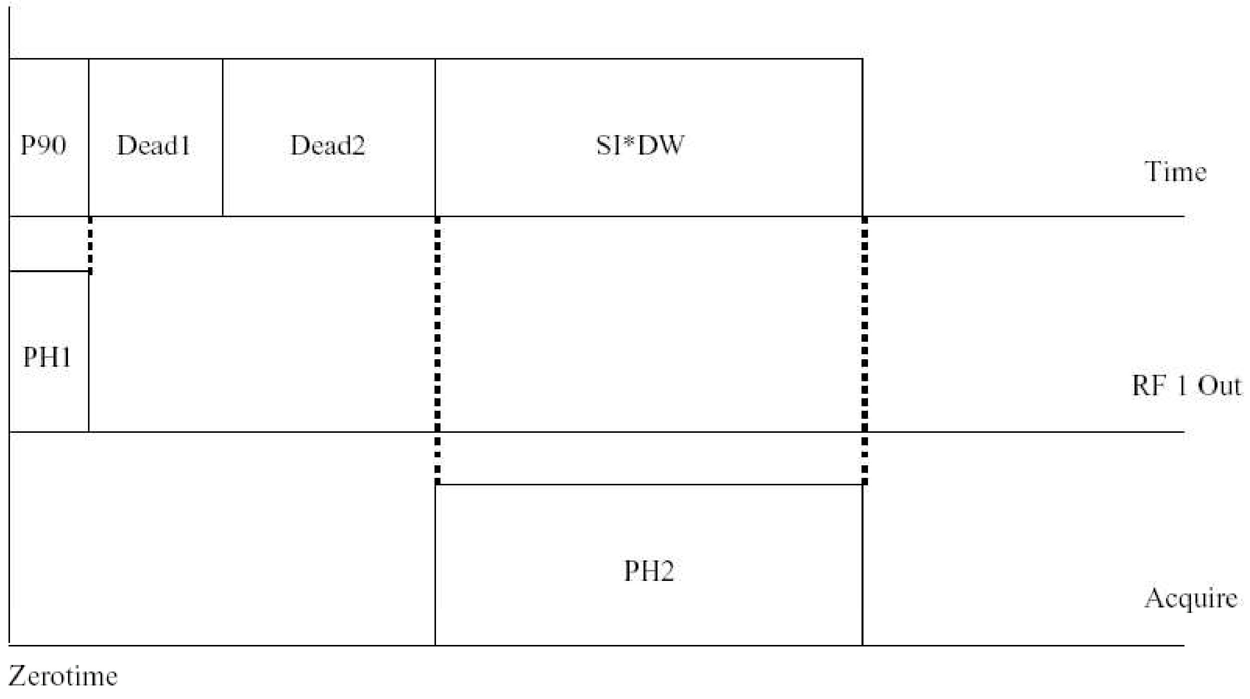
4.3 FID Pulse Sequence

The FID pulse sequence may be used to perform a simple FID experiment using the first RF channel.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



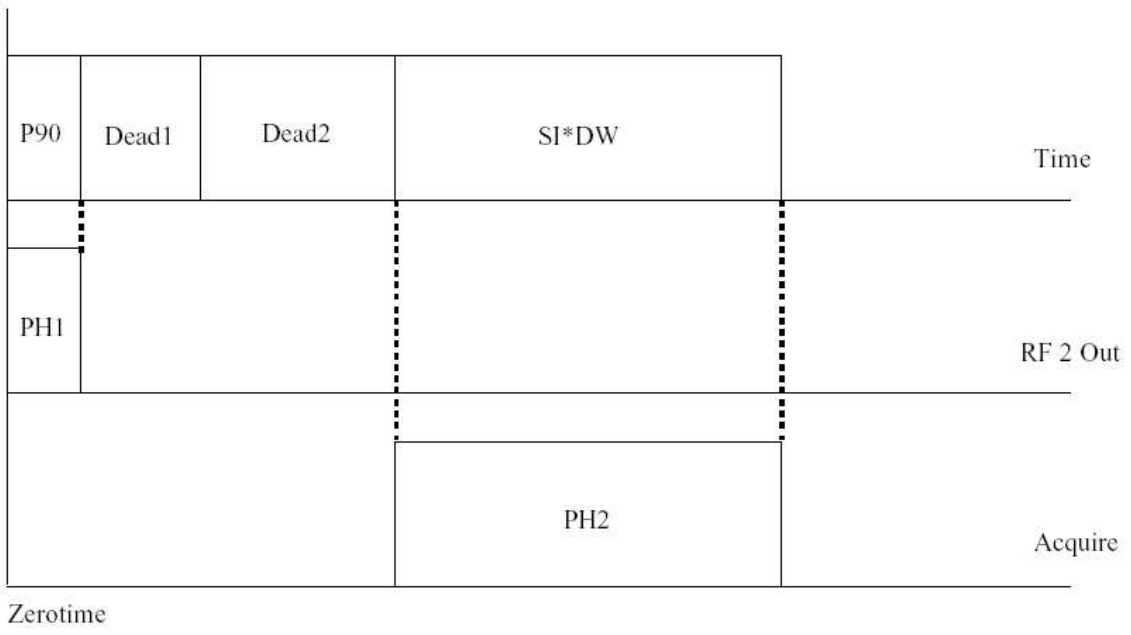
4.4 FIDRF2 Pulse Sequence

This FID pulse sequence may be used to perform a simple FID experiment using the second RF channel.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF2	Spectrometer Frequency Channel 2 (MHz)
O2	Offset from SF2 (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
DS	Dummy Scans
RF2A0	RF Amplitude (%)

Timing Diagram



Notes

This pulse sequence requires that the hardware have a second RF channel to run correctly.

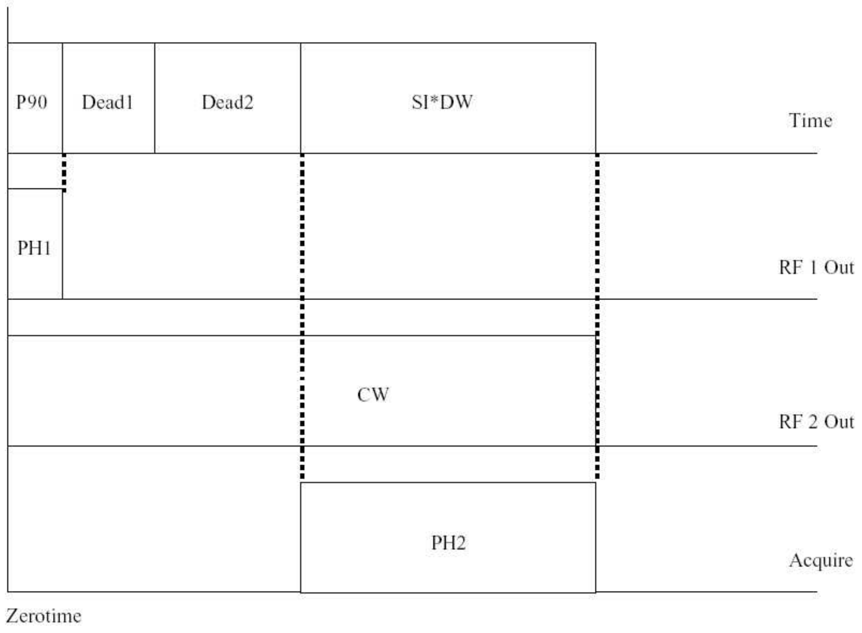
4.5 FIDHCW Pulse Sequence

The FIDHCW pulse sequence may be used to perform a simple FID experiment (first channel) in conjunction with a decoupling pulse (second channel).

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
SF2	Spectrometer Frequency Channel 2 (MHz)
O2	Offset from SF2 (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RF Amplitude Channel 1 (%)
RF2AO	RF Amplitude Channel 2 (decouple) (%)

Timing Diagram



Notes

This pulse sequence requires hardware with two RF channels to run correctly.

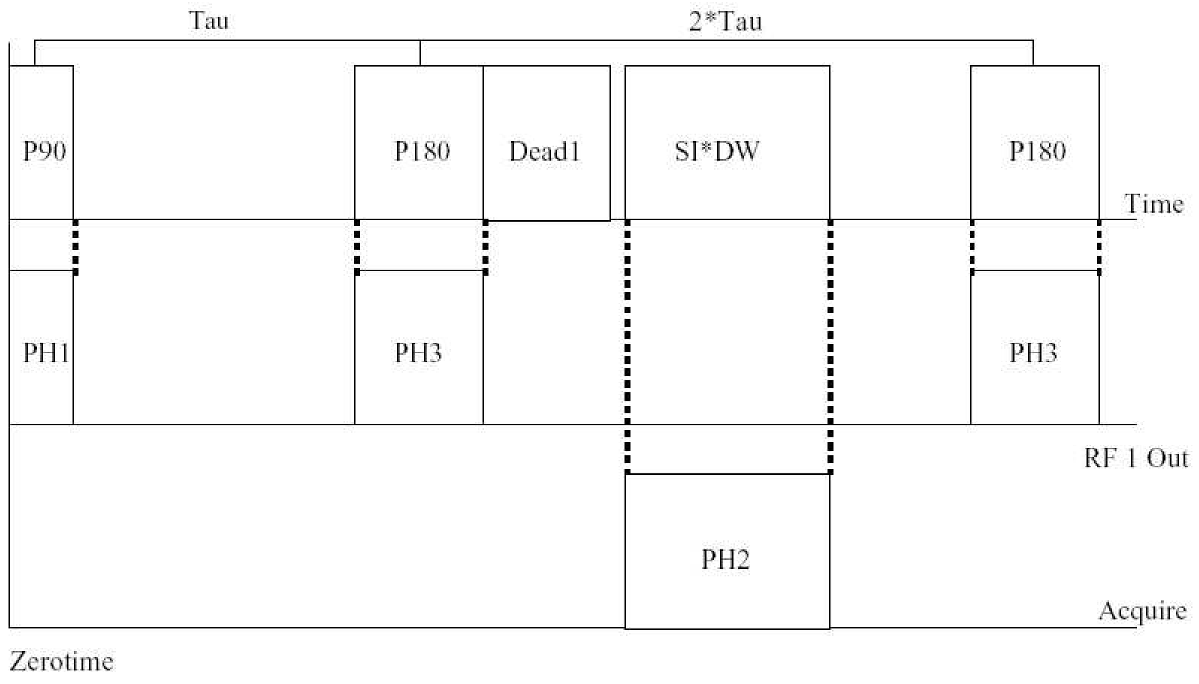
4.6 CPMG Pulse Sequence

The CPMG sequence allows the user to conduct a CPMG experiment using the first RF channel. Users familiar with the MARAN MS-DOS software should note the differences between the new RINMR sequence and the old MS-DOS CPMG sequence, specifically the way in which echoes are acquired.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Points per Echo (points)
NEch	Number of Echoes
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	90-180 Degree Pulse Gap (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 1122)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

The RINMR version of CPMG differs significantly from the MS DOS version. The RINMR pulse sequence acquires ALL echoes. The maximum value of Nech varies depending on instrument configuration.

The process command EVEN may be used to extract only the EVEN echoes from a CPMG data set.

To extract only the even echoes from the data set type:

EVEN
WR filename

When data are acquired with more than 1 point per echo (SI > 1) the points may be averaged into a single point per echo using the PRUNE command.

4.7 CPMGGRAD Pulse Sequence

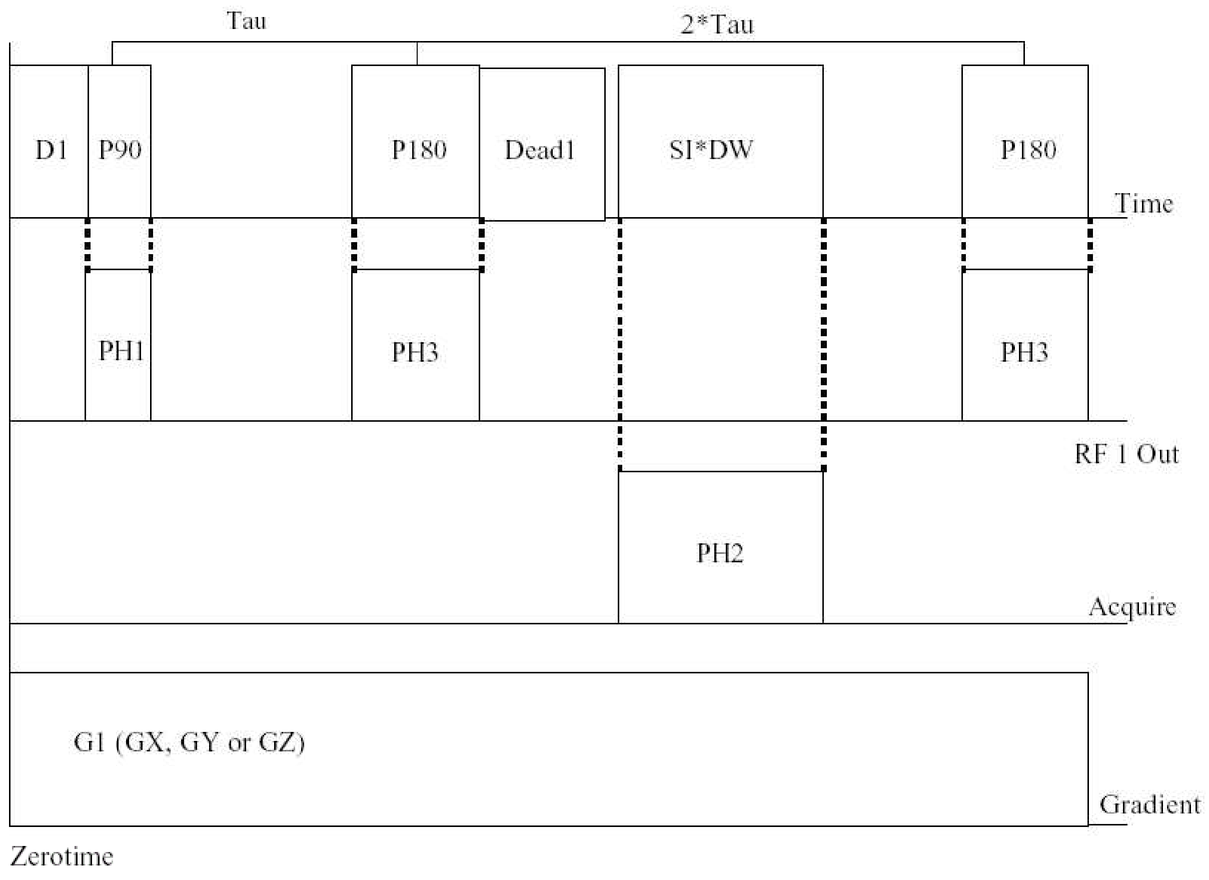
The CPMGGRAD sequence allows the user to conduct a CPMG experiment using the first RF channel. A gradient pulse may be applied during the application of the pulse sequence. Users familiar with the MARAN MS-DOS software should note the differences between the new RINMR sequence and the old MS-DOS CPMGGRAD sequence, specifically the way in which echoes are acquired. Note that CPMGGRAD experiments may generate extremely high gradient duty cycles. Please ensure gradient duty cycles are appropriate before commencing CPMGGRAD experiments. If in any doubt, contact OIMBL for advice.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)

SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Points per Echo (points)
NEch	Number of Echoes
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	90-180 Degree Pulse Gap (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 1122)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)
D1	Gradient Stabilisation Duration (us)
G1	Gradient Pulse Amplitude (-32768 to 32767)
GX	Gradient Pulse Scalar (0 to 32767)
GY	Gradient Pulse Scalar (0 to 32767)
GZ	Gradient Pulse Scalar (0 to 32767)

Timing Diagram



Notes

The RINMR version of CPMG differs significantly from the MS DOS version. The RINMR pulse sequence acquires ALL echoes. The maximum value of NECH varies depending on instrument configuration.

The process command EVEN may be used to extract only the EVEN echoes from a CPMG data set.

To extract only the even echoes from the data set type:

EVEN
WR filename

When data are acquired with more than 1 point per echo (SI > 1) the points may be averaged into a single point per echo using the PRUNE command.

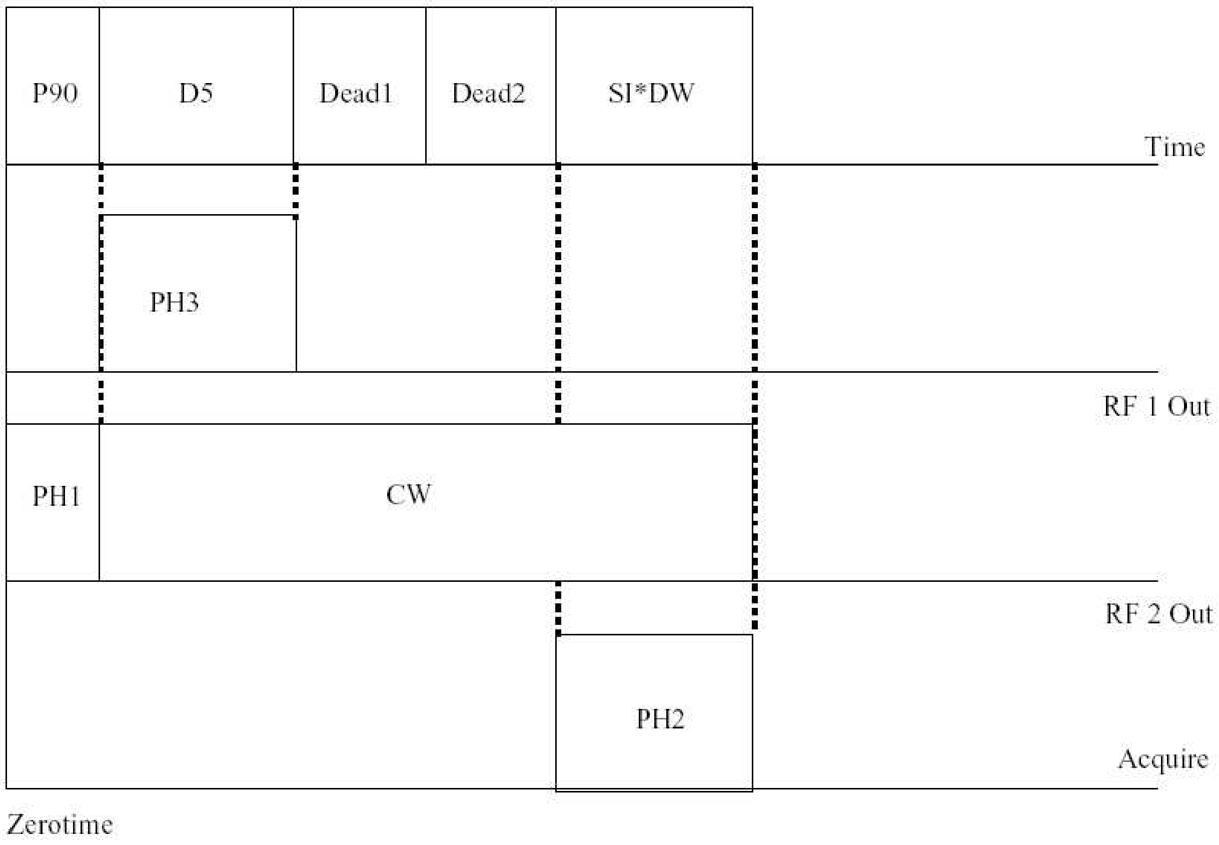
4.8 CPMAS Pulse Sequence

The CPMAS pulse sequence can be used to perform cross polarisation experiments [1].

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
SF2	Spectrometer Frequency Channel 2 (MHz)
O2	Offset from SF2 (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
D5	Contact Duration (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 00112233)
DS	Dummy Scans
RFA0	RF Amplitude Channel 1 (%)
RF2A0	RF Amplitude Channel 2 (%)

Timing Diagram



Notes

This program requires hardware with two RF channels to run correctly.

[1] *Nuclear Magnetic Resonance Spectroscopy*, R K Harris, Pages 150-153.

4.9 INVREC Pulse Sequence

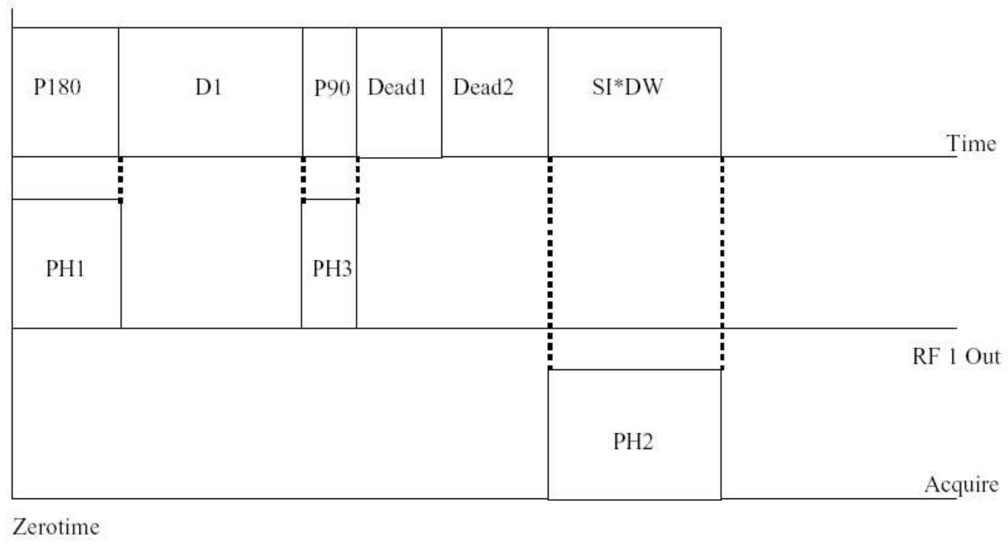
INVREC performs an inversion recovery pulse sequence for the acquisition of T1 weighted data using the first RF channel. A full set of FID's at different inversion recovery times may be acquired and fitted using the T1 script.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)

RD	Relaxation Delay (us)
D1	180-90 Degree Pulse Gap (us)
PH1	90 Degree Pulse Phase List (rec: 1)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RF Amplitude (%)

Timing Diagram



Notes

See the T1 script for more information on measuring T1 time constants using INVREC.

4.10 HAHN Pulse Sequence

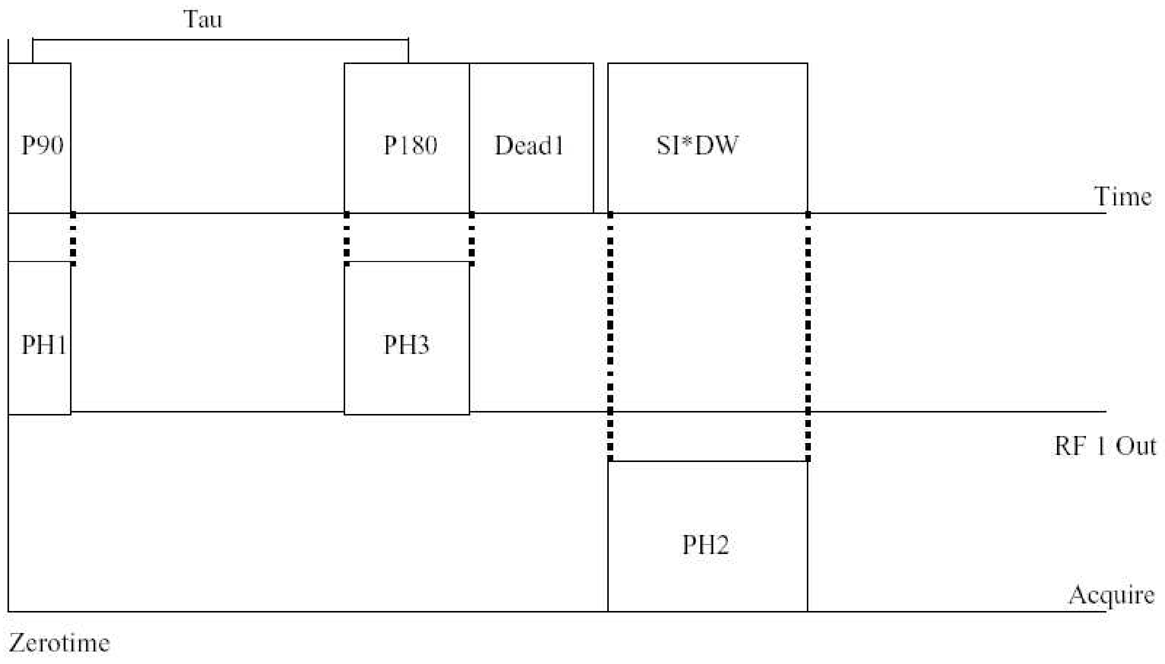
HAHN performs a spin echo experiment using the first RF channel.

Parameters

Parameter	**Description**
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	90-180 Degree Pulse Gap (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)

PH3	180 Degree Pulse Phase List (rec: 0011)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



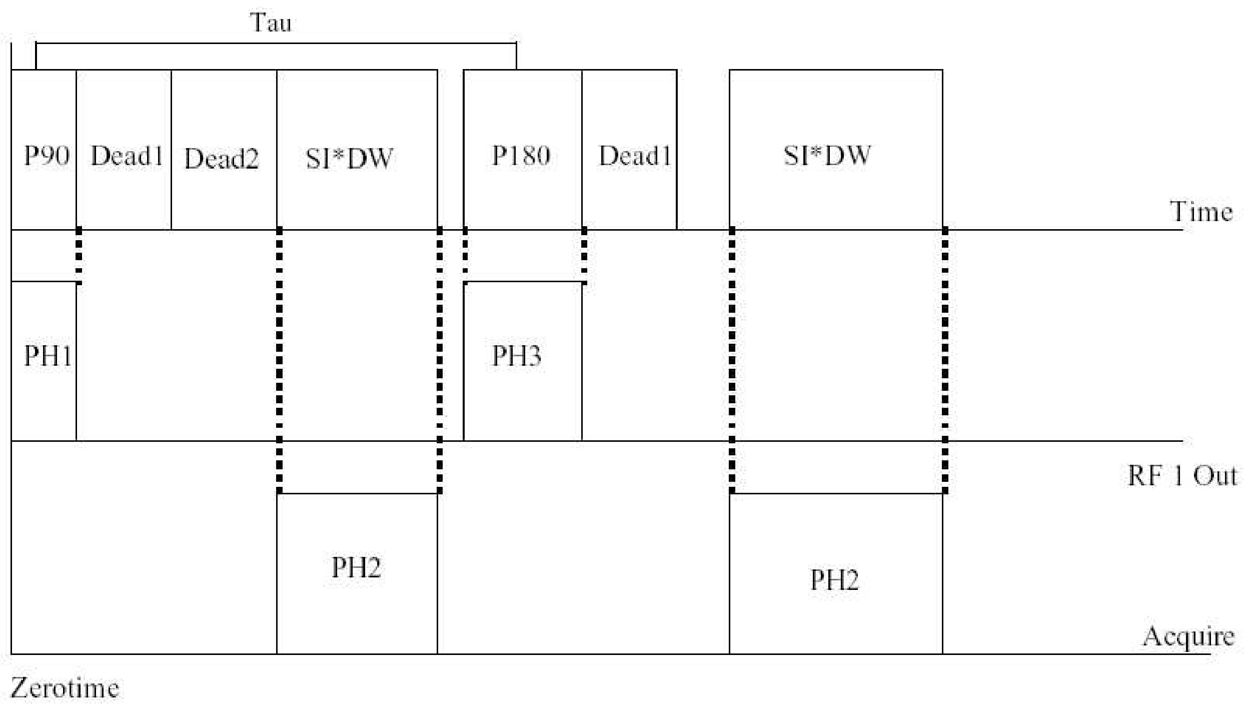
4.11 FID_HAHN Pulse Sequence

FID_HAHN performs a spin echo experiment using the first RF channel. The FID following the initial 90-degree pulse is also acquired.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (FID and Spin Echo) (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	90-180 Degree Pulse Gap (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 0011)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



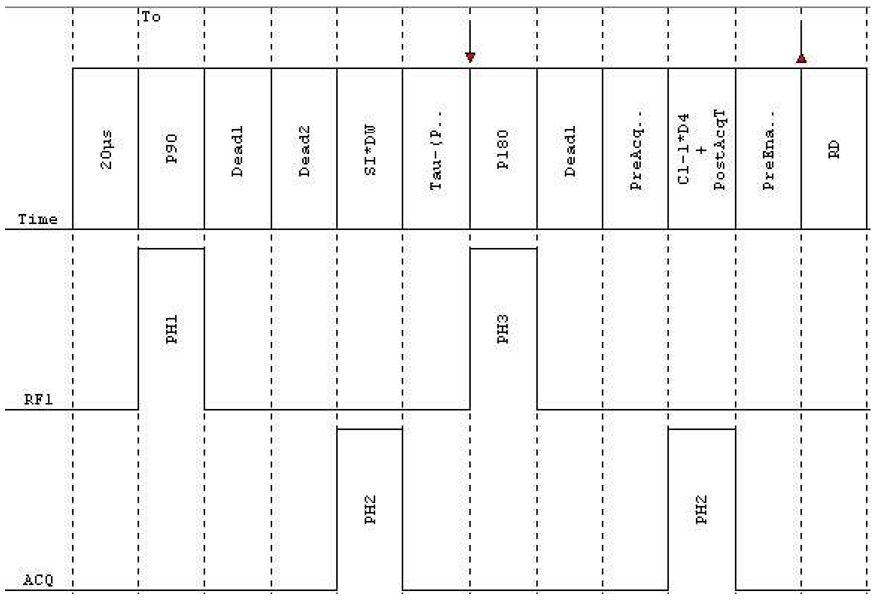
4.12 FIDCPMG Pulse Sequence

FID_HAHN performs a train of spin echoes using the first RF channel. The FID following the initial 90-degree pulse is also acquired.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of FID Acquisition Buffer (points)
C1	Size of Echo Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	90-180 Degree Pulse Gap (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 0011)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

FIDCPMG is capable of generating approximately 32000 data points (FID + echo points). See the Notes for the CPMG sequence for additional information.

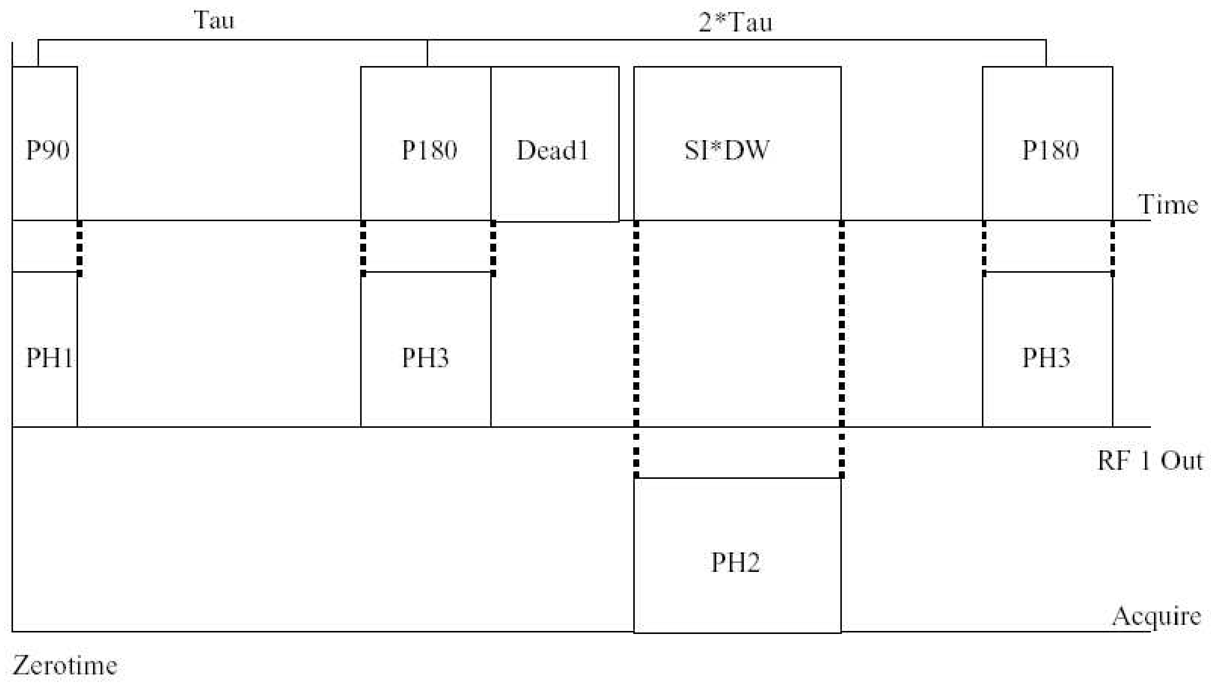
4.13 CPMGF Pulse Sequence

The CPMGF sequence allows the user to conduct a CPMG experiment. CPMGF uses fast RF pulse generation, which leads to a substantial increase in the number of echoes that may be acquired. Note that CMPGF is capable of generating extremely RF duty cycles. Users should ensure that acquisition parameters are set appropriately or damage to the RF probe may occur. If in doubt, please contact OIMBL for advice.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Points per Echo (points)
NEch	Number of Echoes
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	180-90 Degree Pulse Gap (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 1122)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



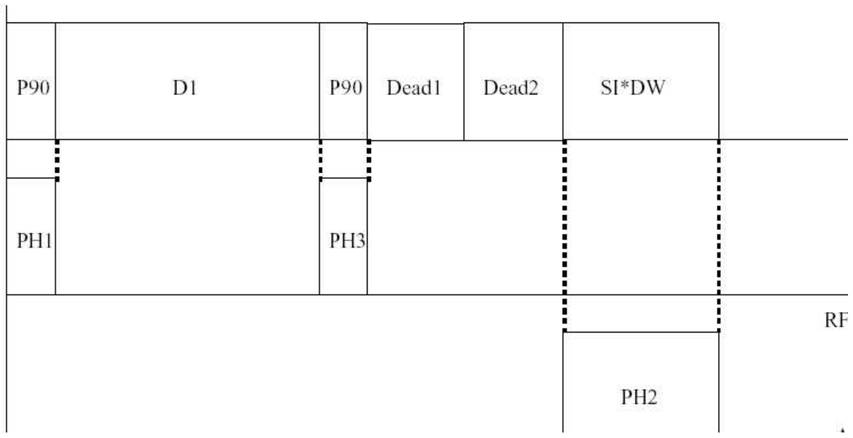
Notes
CPMGF is capable of generating approximately 100000 echoes. See the Notes for the CPMG sequence for additional information.

4.14 SOLID Pulse Sequence
SOLID performs a solid echo experiment.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
D1	90-90 Degree Pulse Gap (us)
PH1	First 90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	Second 90 Degree Pulse Phase List (rec: 1320)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

It should be noted that unlike the HAHN and CPMG experiments, SOLID does not attempt to centre the acquisition on the centre of the solid echo. Acquisition starts immediately following the second 90 degree pulse and its associated dead times. Note also that the pulse gap (D1) runs from the END of the first 90-degree pulse to the START of the second 90-degree pulse.

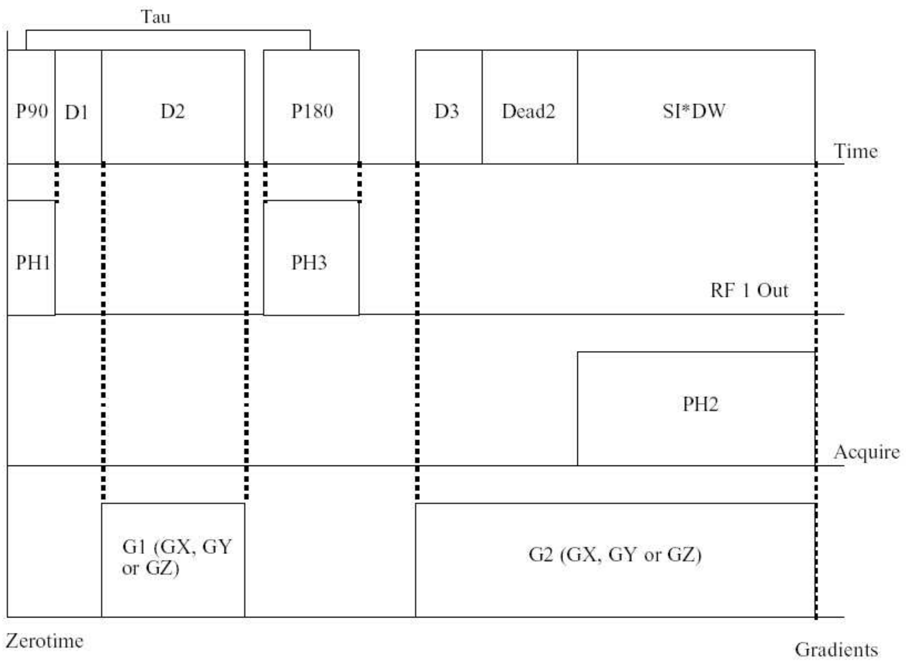
4.15 PROFILE Pulse Sequence

The PROFILE pulse sequence allows the user to conduct a one-dimensional imaging profile.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
D1	Pre Gradient Pulse Duration (us)
D2	Gradient Pulse Duration (dephase) (us)
D3	Pre Acquisition Settle Duration (us)
Tau	90-180 Degree Pulse Gap (us)
G1	Gradient Pulse Amplitude (dephase) (-32768 to 32767)
G2	Gradient Pulse Amplitude (rephase) (-32768 to 32767)
GX	Gradient Pulse Scalar (0 to 32767)
GY	Gradient Pulse Scalar (0 to 32767)
GZ	Gradient Pulse Scalar (0 to 32767)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 0011)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

The gradient strength is given by the expression:

$$\text{Output} = \frac{GN.G1}{32768^2} GMax$$

where GMAX is the maximum output strength of the gradient amplifier, GN (where n may equal X, Y or Z) is the gradient scalar value and G1 is the gradient pulse amplitude. Note that:

- OIMBL spectrometers supplied with a single axis gradient usually have the gradient mapped to the Y channel.
- OIMBL recommends that the DEGAUSS script is run after PROFILE experiments to restore magnet homogeneity.

4.16 DEGAUSS Script

The DEGAUSS script is used to degauss the magnet after the magnetic field gradients have been used.

Notes

OIMBL recommends that the magnet be degaussed following the use of the magnetic field gradients to restore optimum magnet homogeneity.

4.17 WOBBLE Pulse Sequence

The WOBBLE pulse sequence is used to assist in tuning the RF probe.

Parameters

Parameter	Description
SF	Centre Wobble Frequency (MHz)
WW	Wobble Width (MHz)
RFA0	Wobble Pulse Amplitude (0-100%)

Notes

The vertical centre line on the display corresponds to the current value of SF (note SF, not SF+O1). The left hand limit of the RINMR data display corresponds to SF-WW/2 and the right hand limit to SF+WW/2. The horizontal centre line corresponds to a 50ohm match. Note that:

- Some systems must be changed from normal mode to wobble mode before running WOBBLE via the switch on the rear of the magnet box.
- Note that RFA0 may need to be changed from the normal hard RF pulse amplitude (100%) for WOBBLE to function correctly.

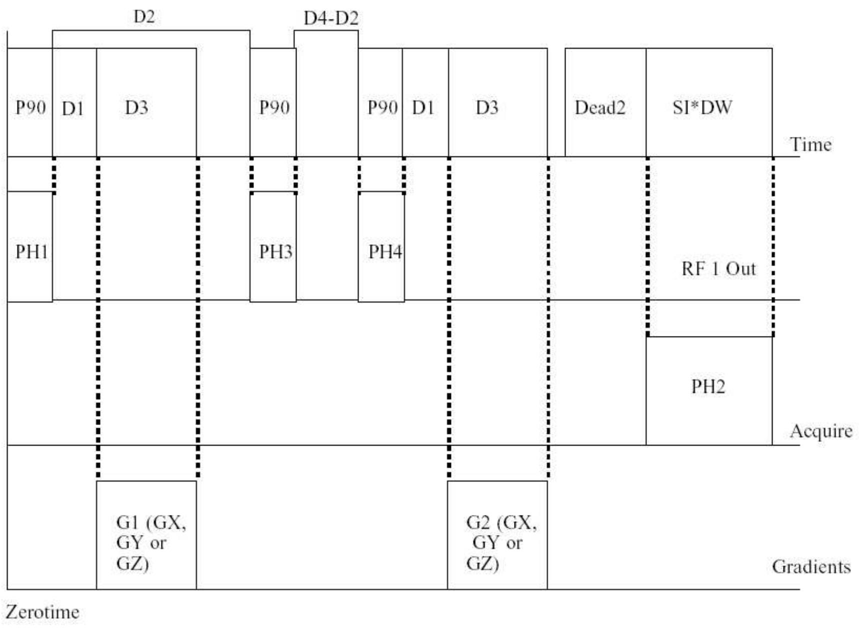
4.18 DIFF Pulse Sequence

The DIFF pulse sequence allows the user to conduct a diffusion experiment. Users should refer to the GETDIFF script for more information on acquiring diffusion data.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
D1	Pre Gradient Delay (us)
D2	Inter 90 Degree Pulse Gap (us)
D3	Gradient Pulse Duration (delta) (us)
D4	Gradient Pulse Separation (DELTA) (us)
G1	Gradient Pulse Amplitude (-32768 to 32767)
GX	Gradient Pulse Scalar (0 to 32767)
GY	Gradient Pulse Scalar (0 to 32767)
GZ	Gradient Pulse Scalar (0 to 32767)
PH1	First 90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	Second 90 Degree Pulse Phase List (rec: 2031)
PH4	Third 90 Degree Pulse Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

The gradient strength is given by the expression:

$$\text{Output} = \frac{GN.G1}{32768^2} GMax$$

where GMAX is the maximum output strength of the gradient amplifier, GN (where n may equal X, Y or Z) is the gradient scalar value and G1 is the gradient pulse amplitude. Note that:

- OIMBL spectrometers supplied with a single axis gradient usually have the gradient mapped to the Y channel.
- OIMBL recommends that the DEGAUSS script is run after DIFF experiments to restore magnet homogeneity.

4.19 DIFFA Pulse Sequence

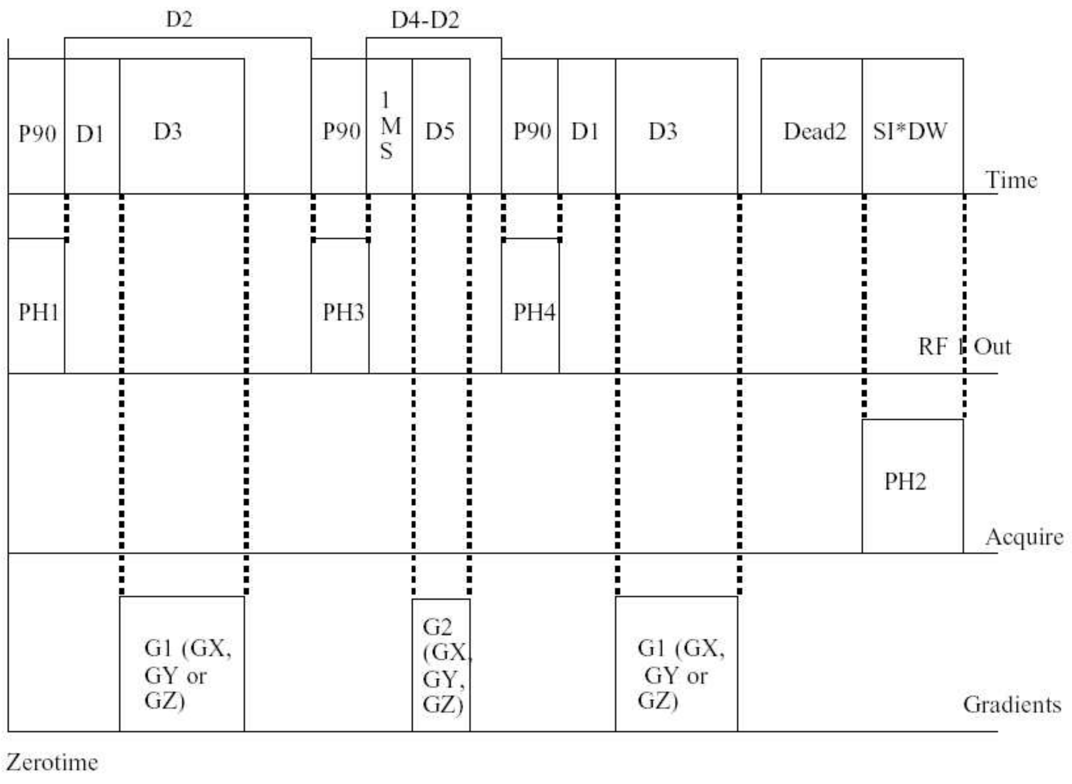
The DIFFA pulse sequence allows the user to conduct a diffusion experiment. DIFFA is an advanced version of the DIFF experiment that incorporates crusher gradients. Users should refer to the GETDIFFA script for more information on acquiring diffusion data.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)

RD	Relaxation Delay (us)
D1	Pre Gradient Delay (us)
D2	Inter 90 Degree Pulse Gap (us)
D3	Gradient Pulse Duration (delta) (us)
D4	Gradient Pulse Separation (DELTA) (us)
D5	Crusher Gradient Pulse Duration (us)
G1	Gradient Pulse Amplitude (-32768 to 32767)
G2	Gradient Pulse Amplitude (-32768 to 32767)
GX	Gradient Pulse Scalar (0 to 32767)
GY	Gradient Pulse Scalar (0 to 32767)
GZ	Gradient Pulse Scalar (0 to 32767)
C1	Acquisition Buffer Extension (points)
PH1	First 90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	Second 90 Degree Pulse Phase List (rec: 2031)
PH4	Third 90 Degree Pulse Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

The gradient strength is given by the expression:

$$\text{Output} = \frac{\text{GN.G1}}{32768^2}\text{GMax}$$

where GMAX is the maximum output strength of the gradient amplifier, GN (where n may equal X, Y or Z) is the gradient scalar value and G1 is the gradient pulse amplitude. Note that:

- OIMBL spectrometers supplied with a single axis gradient usually have the gradient mapped to the Y channel.
- OIMBL recommends that the DEGAUSS script is run after DIFF experiments to restore magnet homogeneity.
- The crusher gradient occurs 1 millisecond after the second 90^o pulse. The C1 parameter can be used to extend the acquisition time to observe the effect of the crusher gradient on the spurious stimulated echoes - do not attempt to alter C1 interactively.

4.20 GETDIFF Script

The GETDIFF script facilitates the measurement of diffusion constants using the DIFF sequence. GETDIFF automatically performs DIFF experiments at a number of different values of D3, which are then saved to disk. GETDIFF should be executed with the syntax .GETDIFF {listname} {filename}, where filename is the prefix of the output files from the individual DIFF experiments and listname is the name of a list file generated with the list editor. For example, if the list file contains the values 1000, 2000, 3000 and 4000, the GETDIFF script will perform 4 DIFF experiments, with a different value of D3. The source code for the GETDIFF script may be found in the script source code directory.

4.21 GETDIFFA Script

The GETDIFFA script facilitates the measurement of diffusion constants using the DIFFA sequence. GETDIFFA automatically performs several DIFFA experiments at a number of different values of D3, which are then saved to disk. GETDIFFA should be executed with the syntax .GETDIFFA {listname} {filename, where filename is the prefix of the output files from the individual DIFFA experiments and listname is the name of a list file generated with the list editor. For example, if the list file contains the values 1000, 2000, 3000 and 4000, the GETDIFFA script will perform 4 DIFFA experiments each with a different value of D3. The source code for the GET_DIFFA script may be found in the script source code directory.

4.22 PGSEMASS Pulse Sequence

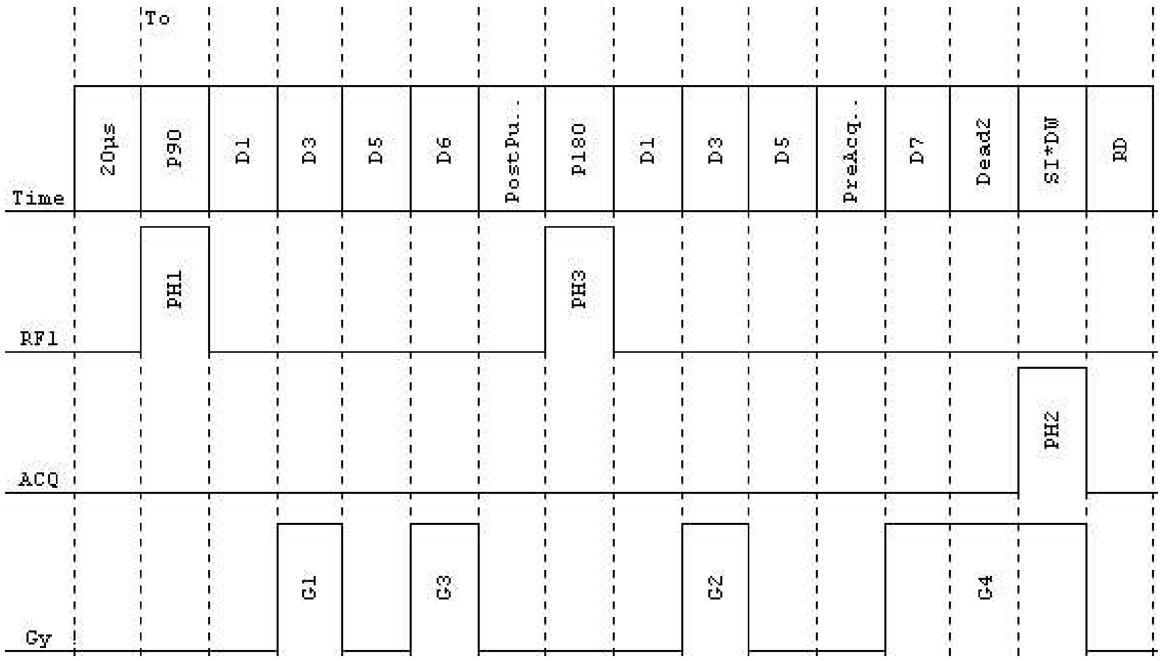
The PGSEMASS pulse sequence allows the user to conduct a diffusion experiment by acquiring individual scans and processing them (Fourier Transform and Magnitude) before the data is accumulated. This procedure is particularly suitable for diffusion experiments on viscous samples using high gradient strength, for which the phase and echo position can fluctuate between scans [2]. The number of individual scans is set by the C1 variable. In such case the standard number of scans NS should be set to 1. Users should refer to the GETPGSEMASS script for more information on acquiring diffusion data.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	Half-echo time duration (ca. Delta) between P90 and P180

D1	Pre Gradient Delay (us)
D3	Gradient Pulse Duration (delta) (us)
D5	Post-diffusion gradient delay (us)
D6	Duration of read-in imaging gradient pulse (us)
D7	Pre-acquisition read-out imaging gradient delay (us)
G1	Diffusion-encoding Gradient Pulse Amplitude (-32768 to 32767)
G2	Diffusion-decoding Gradient Pulse Amplitude (-32768 to 32767)
G3	Read-in imaging Gradient Pulse Amplitude (-32768 to 32767)
G4	Read-out imaging Gradient Pulse Amplitude (-32767 to 32767)
GX	Diffusion-encoding Gradient Pulse Scalar (0 to 32767)
GY	Diffusion-encoding Gradient Pulse Scalar (0 to 32767)
GZ	Diffusion-encoding Gradient Pulse Scalar (0 to 32767)
C1	Number of processed scans
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 0011)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes
OIMBL spectrometers supplied with a single axis gradient usually have the gradient mapped to the Y channel. OIMBL recommends that the DEGAUSS script is run after DIFF experiments to restore magnet homogeneity.

[2] Principle of *Nuclear Magnetic Resonance Microscopy*, P.T. Callagham, Oxford Science Publications, Oxford 1991.

4.23 GETPGSEMASS script

The GETPGSEMASS script facilitates the measurement of diffusion constants using the PGSEMASS sequence when large pulsed field gradients are applied. GETPGSEMASS automatically performs PGSEMASS experiments at a number of different values of G1 (the gradient scaling factor), which are then saved to disk. GETPGSEMASS should be executed with the syntax .GETPGSEMASS {listname} {filename}, where filename is the prefix of the output files from the individual PGSEMASS experiments and listname is the name of a list file generated with the list editor. For example, if the list file contains the values 1000, 2000, 3000 and 4000, the GETPGSEMASS script will perform 4 PGSEMASS experiments, with a different value of G1. The source code for the GETPGSEMASS script may be found in the script source code directory.

4.24 DROPLET Pulse Sequence

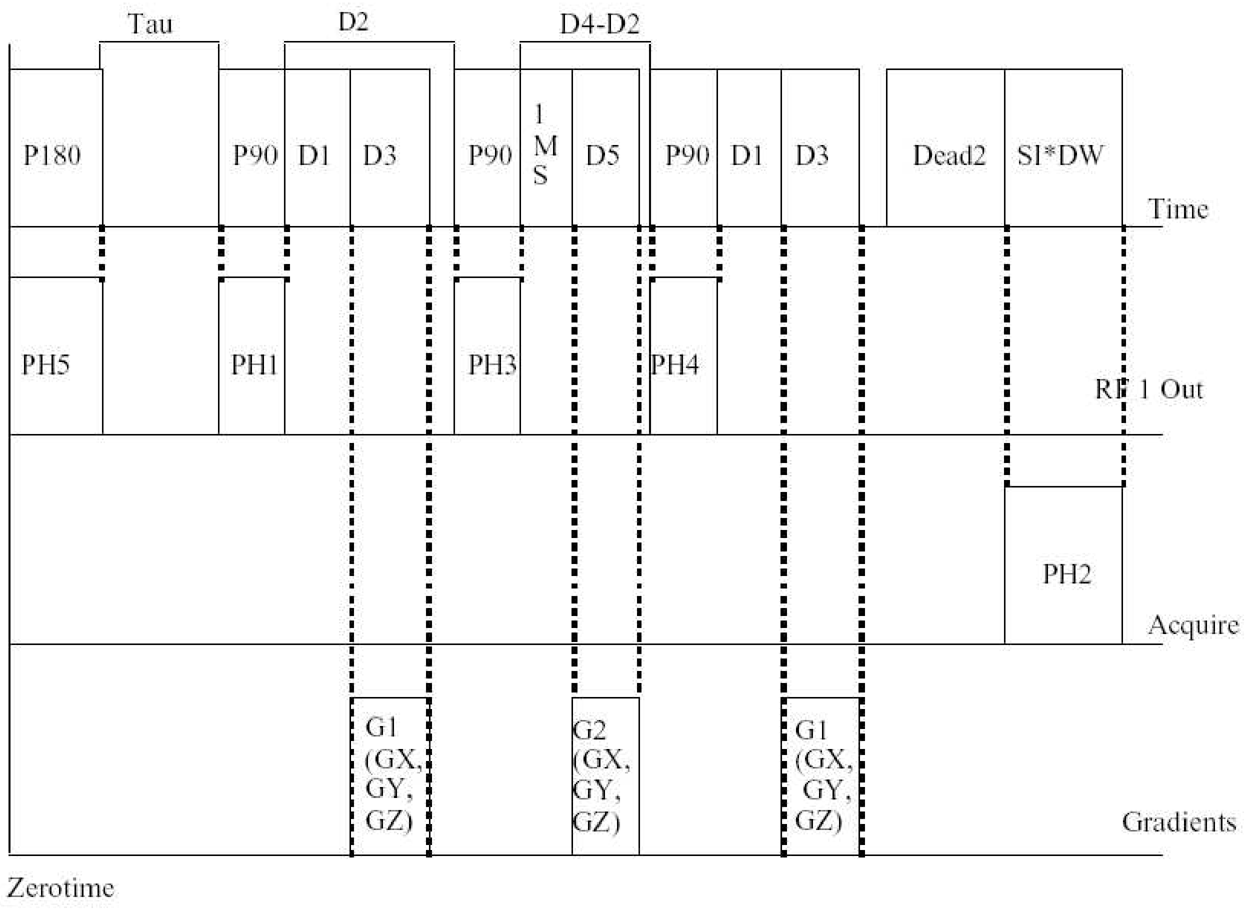
The DROPLET pulse sequence allows the user to conduct droplet size measurements. DROPLET is identical to DIFFA, but incorporates an additional 180^o pulse prior to the diffusion experiment to suppress the signal from the continuous phase.

DROPLET has non-standard parameters to maintain consistency with DIFFA. In particular, the first RF event (the 180 degree pulse) has the phase program PH5 and the suppression time (the time between the 180 degree pulse and the start of the diffusion experiment) is assigned the variable tau. Zero time is defined as the start of the first 90^opulse. Users should refer to the GETDROP and DSD script for more information on acquiring diffusion data for droplet size distribution. Contact OIMBL for details.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	180-90 Suppression Gap (us)
D1	Pre Gradient Delay (us)
D2	Inter 90 Degree Pulse Gap (us)
D3	Gradient Pulse Duration (delta) (us)
D4	Gradient Pulse Separation (DELTA) (us)
D5	Crusher Gradient Pulse Duration (us)
G1	Gradient Pulse Amplitude (-32768 to 32767)
G2	Crusher Gradient Pulse Amplitude (-32768 to 32767)
GX	Gradient Pulse Scalar (0 to 32767)
GY	Gradient Pulse Scalar (0 to 32767)
GZ	Gradient Pulse Scalar (0 to 32767)
C1	Acquisition Buffer Extension (points)
PH1	First 90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	Second 90 Degree Pulse Phase List (rec: 2031)
PH4	Third 90 Degree Pulse Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

The gradient strength is given by the expression:
where GMAX is the maximum output strength of the gradient amplifier, GN (where n may equal X, Y or Z) is the

$$\text{Output} = \frac{GN.G1}{32768^2} GMax$$

gradient scalar value and G1 is the gradient pulse amplitude. Note that:

- OIMBL spectrometers supplied with a single axis gradient usually have the gradient mapped to the Y channel.
- OIMBL recommends that the DEGAUSS script is run after DROPLET experiments to restore magnet homogeneity.
- The crusher gradient occurs 1 millisecond after the second 90^o pulse.
- The C1 parameter can be used to extend the acquisition time to observe the effect of the crusher gradient on the spurious stimulated echoes - do not attempt to alter C1 interactively.

4.25 PGSEMASS Pulse Sequence

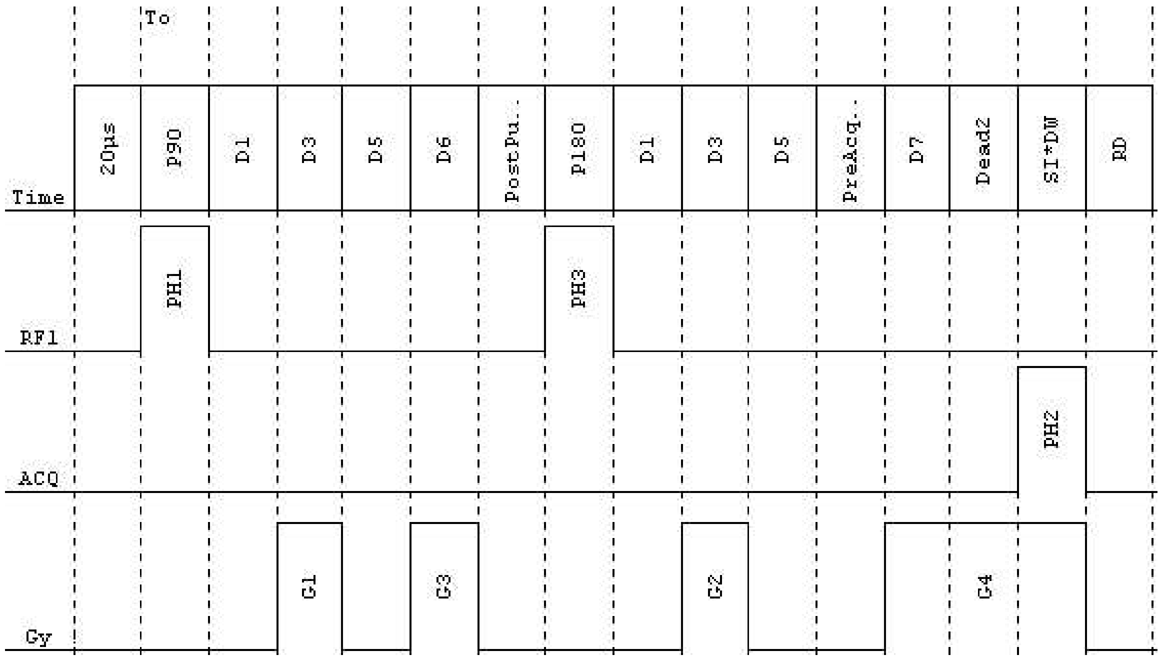
The PGSEMASS pulse sequence allows the user to conduct a diffusion experiment by acquiring individual scans and processing them (Fourier Transform and Magnitude) before the data is accumulated. This procedure is particularly suitable for diffusion experiments on viscous samples using high gradient strength, for which the phase and echo position can fluctuate between scans [2]. The number of individual scans is set by the C1 variable. In such case the standard number of scans NS should be set to 1.

Users should refer to the GETPGSEMASS script for more information on acquiring diffusion data.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
Tau	Half-echo time duration (ca. Delta) between P90 and P180
D1	Pre Gradient Delay (us)
D3	Gradient Pulse Duration (delta) (us)
D5	Post-diffusion gradient delay (us)
D6	Duration of read-in imaging gradient pulse (us)
D7	Pre-acquisition read-out imaging gradient delay (us)
G1	Diffusion-encoding Gradient Pulse Amplitude (-32768 to 32767)
G2	Diffusion-decoding Gradient Pulse Amplitude (-32768 to 32767)
G3	Read-in imaging Gradient Pulse Amplitude (-32768 to 32767)
G4	Read-out imaging Gradient Pulse Amplitude (-32767 to 32767)
GX	Diffusion-encoding Gradient Pulse Scalar (0 to 32767)
GY	Diffusion-encoding Gradient Pulse Scalar (0 to 32767)
GZ	Diffusion-encoding Gradient Pulse Scalar (0 to 32767)
C1	Number of processed scans
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	180 Degree Pulse Phase List (rec: 0011)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



Notes

OIMBL spectrometers supplied with a single axis gradient usually have the gradient mapped to the Y channel. OIMBL recommends that the DEGAUSS script is run after DIFF experiments to restore magnet homogeneity.

[2] Principle of *Nuclear Magnetic Resonance Microscopy*, P.T. Callagham, Oxford Science Publications, Oxford 1991.

4.26 GETPGSEMASS script

The GETPGSEMASS script facilitates the measurement of diffusion constants using the PGSEMASS sequence when large pulsed field gradients are applied. GETPGSEMASS automatically performs PGSEMASS experiments at a number of different values of G1 (the gradient scaling factor), which are then saved to disk. GETPGSEMASS should be executed with the syntax:

.GETPGSEMASS {listname} {filename}

where filename is the prefix of the output files from the individual PGSEMASS experiments and listname is the name of a list file generated with the list editor. For example, if the list file contains the values 1000, 2000, 3000 and 4000, the GETPGSEMASS script will perform 4 PGSEMASS experiments, with a different value of G1. The source code for the GETPGSEMASS script may be found in the script source code directory.

4.27 T1 Script

The T1 script facilitates the measurement of T1 time constants using the INVREC sequence. T1 automatically performs a number of INVREC experiments at a number of different values of D1. T1 should be executed with the syntax .T1 {listname} {filename}where filename is the prefix of the output files from the individual INVREC experiments and listname is the name of a list file generated with the list editor. For example, if the list file contains the values 100, 1000, 10000 and 100000, the T1 script will perform 4 INVREC experiments, substituting the value of D1 each time. Once the program finishes the data are automatically phase rotated and loaded into the T1 fitting program. The T1 data is saved automatically to the filename filename.int. Note that:

- The last entry in the list file for a T1 experiment should be the largest D1 (tau infinity) value.

- When the INVREC sequence runs, it uses the current values of the acquisition parameters (apart from D1). U
- All the acquisition parameters (such as P90, NS, O1 etc) should be set appropriately before executing T1.
- When the T1 fitting program runs, all data files with the prefix filename are used in the T1 calculation.

Further information on how to control the T1 fitting program may be found in the RINMR User manual. The source code for the T1 script may be found in the script source code directory.

4.28 GETDROP Script

The GETDROP script can be used to acquire diffusion data for the measurement of droplet sizes using the DROPLET sequence. GETDROP automatically performs DROPLET experiments at a number of different values of D3, which are then saved to disk. GETDROP should be executed with the syntax:

.GETDROP {listname} {filename}

where filename is the prefix of the output files from the individual DROPLET experiments and listname is the name of a list file generated with the list editor. For example, if the list file contains the values 100, 1000, 10000 and 100000, the GETDROP script will perform 4 DROPLET experiments, with a different value of D3. The source code for the GETDROP script may be found in the script source code directory. Users should note that a more detailed script (DSD script) is now available for routine measurements of droplet size distribution.

4.29 DSD Script

The DSD script facilitates the measurement of droplet sizes using the DROPLET sequence. Refer to method sheet MS002 for more details regarding the droplet size distribution application.

4.30 TRAIN90 Pulse Sequence

The TRAIN90 pulse sequence is designed to allow expert users to accurately set the 90^o pulse length. Non-expert users should use the .AUTOP90 script.

A user specified number of pulses of identical phase and duration are applied to the sample, rotating the magnetisation alternately along the longitudinal axis and the transverse plane. Alternate maxima and minima in signal amplitude are observed when the pulse length is set correctly. As all the 90^o pulses are of identical phase, pulse tip angle errors are cumulative, giving the sequence high sensitivity to 90^o pulse errors.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)
C1	Number of 90 Degree Pulses

4.31 TRAIN180 Pulse Sequence

The TRAIN180 pulse sequence is designed to allow expert users to accurately set the 180^o pulse length. Non-expert users should use the .AUTOP90 script.

A user specified number of pulses of identical phase and duration are applied to the sample. If the 180^o pulse length is specified accurately, the magnetisation is rotated alternately along the positive and negative longitudinal axes. Minima in signal amplitude are observed when the pulse length is set correctly. As all the pulses are of identical phase, pulse tip angle errors are cumulative, giving the sequence high sensitivity to 180^o pulse miss setting.

Parameters

Parameter	Description
P180	180 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
DS	Dummy Scans
RFA0	RF Amplitude (%)
C1	Number of 180 Degree Pulses

4.32 FID_ONE Pulse Sequence

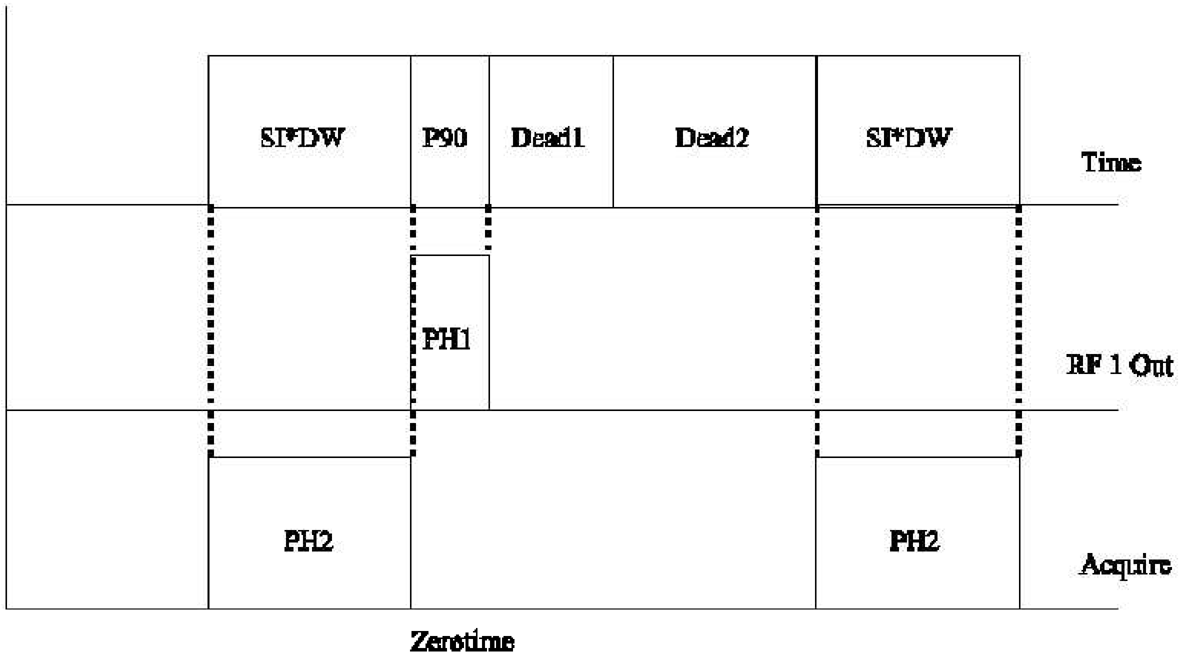
The FID_ONE pulse sequence allows the user to perform an FID experiment. In addition to the usual data acquisition following the 90^o pulse, points are acquired before the 90^o pulse. This allows users to perform a baseline subtraction on data acquired with a single scan, rather than having to perform 4 scans to cancel out any receiver DC offsets.

Parameters

Parameter	Description
P90	90 Degree Pulse (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (before and after the 90 degree pulse, us)
SI	Size of Acquisition Buffer (points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)

DS	Dummy Scans
RFA0	RFA0 RF Amplitude (%)

Timing Diagram



4.33 LISTGEN Script

LISTGEN allows users to generate linear spaced, logarithmic spaced or geometric spaced lists for use with scripts such as T1, GETDIFF and GETDROP. LISTGEN should be executed with the following syntax:

.LISTGEN

The steps involved in the list generators are as follows:

1. When LISTGEN is executed it will prompt for a name to save the list to (lists should be saved in the C:\Program Files\RINMR\list directory).

2. LISTGEN will then prompt for values that will characterise the list.

3. Finally, LISTGEN will prompt for a list type. The types are as follows:

   - ***Linear***:- values will be spaced linearly between the start and end values according to the following equation:

$$T_i = T_0 + i\frac{(T_0 - T_n)}{(n-1)}$$

   - ***Logarithmic***:- values will be spaced logarithmically between the start and end values according to the following equation:

$$T_i = e^{\frac{(\ln(T_n(n-i-1)) + \ln(iT_0))}{(n-1)}}$$

- ***Geometric***:- values will be spaced geometrically between the start and end values according to the following equation:

$$T_i = T_0 a^i \qquad \text{where } a = \left(\frac{T_n}{T_0}\right)^{\left(\frac{1}{n-1}\right)}$$

where T_i is the ith value in the list, T_0 is the first value, T_n is the last value and n is the total number of values in the list.

4.34 PROBECHANGE Script

The PROBECHANGE script allows the user to configure a text file that contains details of various probes that may be supplied with the MARAN/MARAN Ultra instruments. When the PROBECHANGE script is executed, the user is presented with a list of probes and a choice may be made. Following the selection, the system parameters P90 and Dead1 are automatically updated to the values specified in the probe configuration file.

Editing the Probe Configuration File

The probe configuration file is an ASCII text file that should reside in the \RINMR\BIN directory and have the filename probes.txt. The file should have a list of probe names (as they will appear in the selection window) followed by the P90 and Dead1 for each particular probe. Note that a P90 and Dead1 MUST be specified for each probe and that a maximum of 4 probes may be specified.

Example of Probe Configuration File

```
18 mm VT Probe With Gradients
12.2
7.0
10 mm VT Probe
2.3
4.3
26 mm Absolute Probe
15.0
12.0
```

4.35 GRADCALIB Script

The GRADCALIB script allows the user to calibrate the strength of the magnetic field gradients and also estimate the thickness of slices used in imaging sequences. The gradient calibration is performed using the PROFILE pulse sequence and the slice thickness is calibrated using the SL90XYZ pulse sequence. When GRADCALIB is executed the user is presented with two options:

Calibrate Gradient Strength - If this box is checked the software will prompt for values to allow the user to calibrate the gradient strength.

Enter the width of the 1D profile in Hz (this can be measured using the crosshair, note that after the FT command the x axis of RINMR converts to frequency units), the length of the sample in cm, the gradient scalar value used (either GX, GY and GZ) and the value of G2 used to acquire the profile and the gradient amplifier maximum output current (nominally 55A for Crown Macrotechs). The gradient strength is displayed in various units.

Calibrate slice thickness - If this box is checked the software will prompt for values to allow the user to calibrate the slice excitation thickness. Note that if both boxes are checked the software will automatically use the gradient strength calculated in the gradient calibration for calculation of the slice excitation thickness.

Enter the width of the excitation slice in Hz, the slice gradient strength, the slice gradient scalar and the maximum gradient strength of the slice select gradient (calculated from (i)). The slice excitation thickness is displayed in cm.

Additional Information

The calibrated values for the gradient strength in T/m/A (Tesla per metre per Amp) can be inserted into the RINMR.ini file in the \RINMR\bin directory. More information on calibrating the gradients using a one-dimensional profile can be found in the MARAN Ultra Non-Expert User Manual.

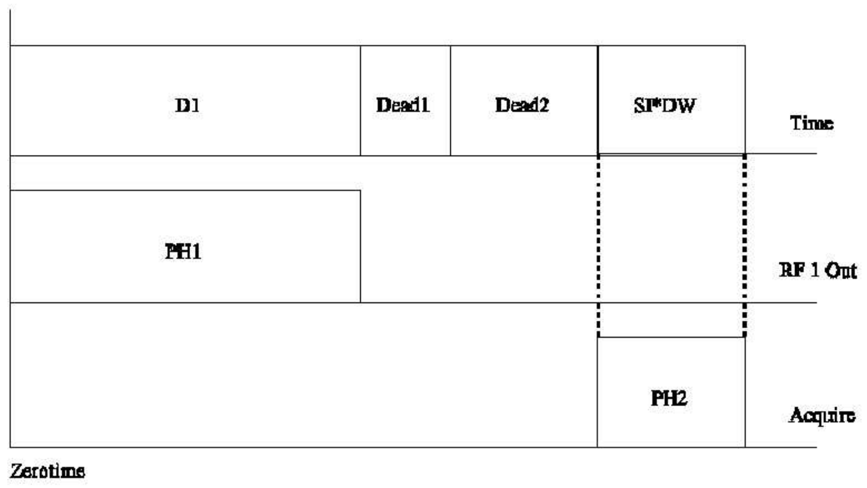
4.36 SETT1RHO Pulse Sequence

The SETT1RHO pulse sequence can be used to set the amplitude of a spin lock pulse for use with the T1RHO pulse sequence. Note that SETT1RHO pulse sequence requires the T1RHO option to function correctly.

Parameters

Parameter	Description
D1	Spin Lock Pulse Duration (us)
Dead1	Probe Dead Time (us)
Dead2	Receiver Dead Time (us)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (us)
SI	Size of Acquisition Buffer (before and after the 90 degree pulse, points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	Spin Lock Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
DS	Dummy Scans
RFA1	RFA1 Spin Lock RF Amplitude (%)

Timing Diagram



Notes

The SETT1RHO pulse sequence can be run to calculate the T_1rho pulse illumination bandwidth and the magnetic field strength. It allows the user to conduct an FID experiment with a pulse of B_1 amplitude identical to that of the T1RHO pulse and to set the 90º pulse length accordingly.

The illumination bandwidth is defined by the equation:

$f=1/(4*t_p)$

where f is the illumination bandwidth (Hz) and t_p is the 90º pulse duration (D3, seconds). For example, a 90º pulse of 6μs has a 40 KHz illumination bandwidth. The following expression should be used to calculate the B_1 amplitude of the 90º pulse:

$\pi/2=2*\pi*\gamma*B_1*t_p$

$B_1=1/(4*\gamma*t_p)$

where B_1 is the magnetic field strength of the 90º pulse and γ is the magnetogyric ratio. B_1 is approximately 10G for a 90º pulse of duration 6μs at the resonance frequency of hydrogen protons. Users should realise that exceeding the recommend duty cycle could lead to damage to the RF probe.

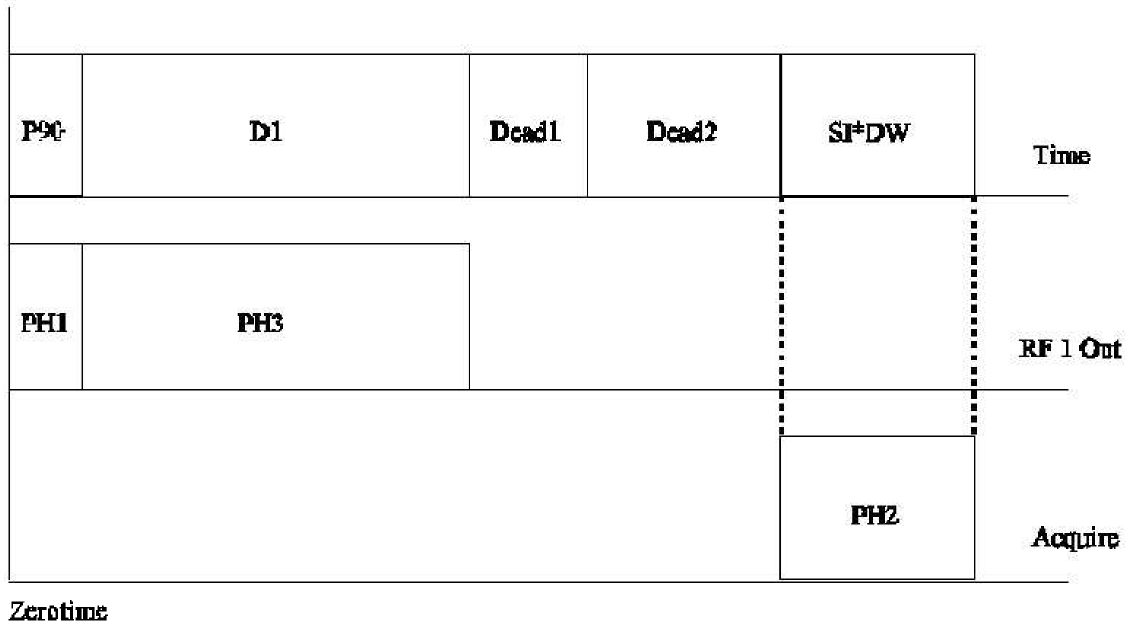
4.37 T1RHO Pulse Sequence

The T1RHO pulse sequence can be used to perform T_1rho measurements. Note that the T1RHO pulse sequence requires the T_1rho hardware option to function correctly. The GETT1RHO script can be used to facilitate the acquisition of T_1rho data.

Parameters

Parameter	Description
P90	90 Degree Pulse Duration (μs)
D1	Spin Lock Pulse Duration (μs)
Dead1	Probe Dead Time (μs)
Dead2	Receiver Dead Time (μs)
SF	Spectrometer Frequency (MHz)
O1	Offset from SF (Hz)
FW	Filter Width (Hz)
DW	Dwell Time (μs)
SI	Size of Acquisition Buffer (before and after the 90 degree pulse, points)
NS	Number of Scans
RG	Receiver Gain (%)
RD	Relaxation Delay (us)
PH1	90 Degree Pulse Phase List (rec: 0213)
PH2	Receiver Phase List (rec: 0213)
PH3	Spin Lock Pulse Phase List (rec: 3102)
DS	Dummy Scans
RFA0	90 Degree Pulse RF Amplitude (%)
RFA1	Spin Lock Pulse RF Amplitude (%)

Timing Diagram



Notes

The T1RHO pulse sequence uses the fast RF pulse generation function (FRF) to ensure that there is no delay between the 90° and the spin locking pulses. The maximum duration of the spin locking pulse is hardware limited to 1 second.

4.38 GETT1RHO Script

The GETT1RHO script facilitates the measurement of T_1rho time constants using the T1RHO pulse sequence. GETT1RHO automatically performs a number of spin locking experiments at a number of different values of D1 (spin lock pulse duration). GETT1RHO should be executed with the syntax .GETT1RHO {listname} {filename}, where filename is the prefix of the output files from the individual T1RHO experiments and listname is the name of a list file generated with the list editor. For example, if the list file contains the values 100, 1000, 10000 and 100000, the GETT1RHO script will perform 4 T1RHO experiments, substituting the value of D1 each time. Note that:

- The last entry in the list file for a GETT1RHO experiment should be the largest D1 (tau infinity) value.
- When the GETT1RHO sequence runs, it uses the current values of the acquisition parameters (apart from D1). Users should ensure that all the acquisition parameters (such as P90, NS, O1, RFA0 and RFA1 etc) are set appropriately before executing GETT1RHO.

The source code for the GETT1RHO script may be found in the script source code directory.

4.39 TD Script

The TD script starts the RI Timing Diagram software and displays the currently loaded pulse sequence in a graphic format. The RINMR User Manual provides more information on the RI Timing Diagram software. Simply type .TD on the RINMR command line to display the currently loaded pulse sequence in graphic form.

4.40 Imaging sequences and scripts

The MARAN Ultra is also supplied with a list of imaging sequences. Refer to the MARAN Ultra Imaging manual for their description and use.

Chapter 5 The RINMR File Format

5.1 Introduction

This chapter describes the file format definition for the binary RINMR *.RiDat and *.RiImage files.

5.2 RINMR File Format Definition

The RINMR data files consist of two main parts. The file starts with the header, which contains a list of all the parameters used to collect the data. The header is followed by the data itself.

5.3 Header

The header itself is divided into 4 sections. Each section contains a different type of parameter. Each section is a multiple of 512 bytes. Not every byte in each section is used. Any unused bytes are all set to zero. This allows new parameters to be added at a later date without changing the header size.

5.3.1 Identification Section

The first section currently has the following parameters defined:

```
Magic          : INTEGER;
FileVersion    : INTEGER;
Section1Size   : INTEGER;
Section2Size   : INTEGER;
Section3Size   : INTEGER;
Section4Size   : INTEGER;
Comment        : ARRAY[0..128] OF CHAR;
```

"Magic" is set to 190955 to all NMR data files. The parameters "Section#Size" where # is 1, 2, 3, or 4 define the size of each of the 4 sections in the header. These values are in bytes. Defining the size of each section like this will allow any section to be increased in future versions of the software without affecting any existing programs that correctly take account of the size of each section.

5.3.2 System Parameters

This section contains the system parameters (see the RINMR User Manual for a definition of which parameters are system parameters).

5.3.3 Application Parameters

This section contains the application parameters (see the RINMR User Manual for a definition of which parameters are application parameters).

5.3.4 Processing Parameters

This section contains the process parameters (see the RINMR User Manual for a definition of which parameters are process parameters).

5.4 Data Files

The data follows the header. The NMR data consists of complex data points. In the data files the time value for each data point relative to a starting point defined in the pulse program (the ZeroTime label) is also stored.

```
DataA  : SINGLE;
DataB  : SINGLE;
Time   : DOUBLE;
```

DataA and DataB are dimensionless and Time is in 0.1μs units.

5.5 Image Files

Image files (.RiImage) have an identical header to data files (.RiDat), except that the FileVersion field is set to 1 instead of 0. Immediately following the header are the time values. The number of time values is defined in the parameter SI. All the time values are stored as DOUBLE (8 bytes). After the time values there is the actual NMR data. This is stored as pairs of SINGLEs (4 bytes). For multidimensional images each image is stored sequentially.

5.6 Pascal Header Definition

The following code defines the RINMR data file header. All INTEGERSs are 4 bytes long. Pascal strings store the length of the string in the first byte and there is no terminating NULL character. All variables start at a 4-byte boundary, which means there are sometimes extra bytes which are used to pad out to the next 4 bytes. This usually only happens at the end of string variables.

```
CONST
US    = 1E-6;
MS    = 1E-3;
S     = 1;

MAX_SHAPE_LENGTH = 16;

TYPE
PhaseList        = STRING[128];
PhaseListPtr     = ^PhaseList;

StringList       = array[0..128] of Char;
StringListPtr    = ^StringList;

ShapeString      = ARRAY[0..MAX_SHAPE_LENGTH] OF CHAR;

VAR
// The order of the variables in these blocks must never change. Any new
// variables must be added just in front of the end markers.

// Identification Parameters

Magic            : INTEGER;
FileVersion      : INTEGER;
Section1Size     : INTEGER;
Section2Size     : INTEGER;
Section3Size     : INTEGER;
Section4Size     : INTEGER;
Comment          : StringList;  //ARRAY[0..128] OF CHAR;

IdentEndMark     : INTEGER;     // This marks the end of the Identification
                                Block
                                // This variable must always be the last
                                one.
                                // System Parameters

      Dead1            : SINGLE; // 0
      Dead2            : SINGLE;
      P90              : SINGLE;
      P180             : SINGLE;
```

```
    SF                : DOUBLE;
    O1                : DOUBLE;

    Dummy1            : INTEGER; // This was added by mistake but must NOT
                                    Be removed. It can however be replaced.
    MultReg           : INTEGER;      // 10
    PhaseTwiddle      : INTEGER;
    ChanAOffset       : INTEGER;
    ChanBOffset       : INTEGER;
    ExtAPhaseTrim     : INTEGER;
    ExtAAmpTrim       : INTEGER;
    ExtBPhaseTrim     : INTEGER;
    ExtBAmpTrim       : INTEGER;
    IntAAmpTrim       : INTEGER;
    IntBAmpTrim       : INTEGER;
    PhaseTrim0        : INTEGER;
    AmpTrim0          : INTEGER;      // 20
    PhaseTrim180      : INTEGER;
    AmpTrim180        : INTEGER;
    PhaseTrim90       : INTEGER;
    AmpTrim90         : INTEGER;
    PhaseTrim270      : INTEGER;
    AmpTrim270        : INTEGER;

    SF2               : DOUBLE;
    O2                : DOUBLE;

    RF2MultReg        : INTEGER;      // 31
    RF2PhaseTwiddle   : INTEGER;
    RF2ChanAOffset    : INTEGER;
    RF2ChanBOffset    : INTEGER;
    RF2ExtAPhaseTrim  : INTEGER;
    RF2ExtAAmpTrim    : INTEGER;
    RF2ExtBPhaseTrim  : INTEGER;
    RF2ExtBAmpTrim    : INTEGER;
    RF2IntAAmpTrim    : INTEGER;
    RF2IntBAmpTrim    : INTEGER;      // 40
    RF2PhaseTrim0     : INTEGER;
    RF2AmpTrim0       : INTEGER;
    RF2PhaseTrim180   : INTEGER;
    RF2AmpTrim180     : INTEGER;
    RF2PhaseTrim90    : INTEGER;
    RF2AmpTrim90      : INTEGER;
    RF2PhaseTrim270   : INTEGER;
    RF2AmpTrim270     : INTEGER;

    SF3               : DOUBLE;
    O3                : DOUBLE;
    RF3MultReg        : INTEGER;      // 53
    RF3PhaseTwiddle   : INTEGER;
    RF3ChanAOffset    : INTEGER;
    RF3ChanBOffset    : INTEGER;
```

```
     RF3ExtAPhaseTrim : INTEGER;
     RF3ExtAAmpTrim   : INTEGER;
     RF3ExtBPhaseTrim : INTEGER;
     RF3ExtBAmpTrim   : INTEGER;      // 60
     RF3IntAAmpTrim   : INTEGER;
     RF3IntBAmpTrim   : INTEGER;
     RF3PhaseTrim0    : INTEGER;
     RF3AmpTrim0      : INTEGER;
     RF3PhaseTrim180  : INTEGER;
     RF3AmpTrim180    : INTEGER;
     RF3PhaseTrim90   : INTEGER;
     RF3AmpTrim90     : INTEGER;
     RF3PhaseTrim270  : INTEGER;
     RF3AmpTrim270    : INTEGER;      // 70

     RF1QuadTrim      : INTEGER;
     RF2QuadTrim      : INTEGER;
     RF3QuadTrim      : INTEGER;

SysEndMark            : INTEGER;      // This marks the end of the
                                      Parameter Block
                                      // This variable must always be the
                                      last one.

// Application Parameters

     Points           : INTEGER;      // 0

     Dwell            : SINGLE;

     P1               : SINGLE;
     P2               : SINGLE;
     P3               : SINGLE;
     P4               : SINGLE;
     P5               : SINGLE;

     RD               : SINGLE;
     Tau              : SINGLE;
     D1               : SINGLE;
     D2               : SINGLE;       // 10
     D3               : SINGLE;
     D4               : SINGLE;
     D5               : SINGLE;

     Scans            : INTEGER;
     Filter           : SINGLE;

     PH1              : PhaseList;
     PH2              : PhaseList;
     PH3              : PhaseList;
     PH4              : PhaseList;
     PH5              : PhaseList;
     Gain                : SINGLE;

     NumECHOES           : INTEGER;

     SweepWidth          : DOUBLE;
     Attenuation         : INTEGER;
     Bessel              : DOUBLE;
     Butterworth         : DOUBLE;
     SeqName             : ARRAY[0..31] OF CHAR;
     RFA0                : SINGLE;
     RFA1                : SINGLE;
     RFA2                : SINGLE;
     RFA3                : SINGLE;
     RFA4                : SINGLE;
     RFA5                : SINGLE;
     RF2A0               : SINGLE;
     RF2A1               : SINGLE;
     RF2A2               : SINGLE;
     RF2A3               : SINGLE;
     RF2A4               : SINGLE;
     RF2A5               : SINGLE;
     WobbleWidth         : SINGLE;
     C1                  : INTEGER;
     C2                  : INTEGER;
     C3                  : INTEGER;
     C4                  : INTEGER;
     C5                  : INTEGER;

     GX                  : INTEGER;
     GY                  : INTEGER;
     GZ                  : INTEGER;
     G1                  : INTEGER;
     G2                  : INTEGER;
     G3                  : INTEGER;
     G4                  : INTEGER;
     G5                  : INTEGER;
     G6                  : INTEGER;
     G7                  : INTEGER;
     G8                  : INTEGER;
     G9                  : INTEGER;

     MAC1                : SINGLE;
     MAC2                : SINGLE;

     SH1                 : ShapeString;
     SH2                 : ShapeString;
     SH3                 : ShapeString;
     SH4                 : ShapeString;
     SH5                 : ShapeString;

     DS                  : INTEGER;
     NA                  : INTEGER;
```

```
      IG1                : INTEGER;
      IG2                : INTEGER;
      IG3                : INTEGER;
      IG4                : INTEGER;
      IG5                : INTEGER;
      IG6                : INTEGER;
      IG7                : INTEGER;
      IG8                : INTEGER;
      IG9                : INTEGER;
      DimX               : INTEGER;
      DimY               : INTEGER;
      DimZ               : INTEGER;
      DimC               : INTEGER;
      ImageEchoes        : INTEGER;
      ImageSlices        : INTEGER;

AppEndMark               : INTEGER;      // This marks the end of the
                                         Application Block
                                         // This variable must always be the
                                         last one.

// Processing Parameters

      ProcessFlags       : INTEGER;
      ProcessDummy1      : INTEGER;
      ProcessDummy2      : INTEGER;
      ProcessDummy3      : INTEGER;
      ProcessDummy4      : INTEGER;
      ProcessDummy5      : INTEGER;
      ProcessDummy6      : INTEGER;
      ProcessDummy7      : INTEGER;
      ProcessDummy8      : INTEGER;
      ProcessDummy9      : INTEGER;

      LB                 : SINGLE;
      PA                 : SINGLE;
      PB                 : SINGLE;
      DP                 : Single;
      SMP                : INTEGER;

      ProcEndMark        : INTEGER;      // This marks the end of the
                                         Processing Block.
                                         // This variable must always be the
                                         last one.
```

5.7 IDL Image Data Loader

This section contains an example procedure that allows the user to load an image file (.RIImage) into IDL.

```
PRO Read2DFile,Filename,err               ; Filename passed to the procedure
OpenR, Unit, FileName,/Get_LUN,Error=Err ; Open the image file
IF Err NE 0 THEN RETURN                   ; Return if file open fails
```

```
ParBuff=LonArr(10) ; Read in First Few Bytes of Header to Get Size of
sections
Point_Lun,Unit,0
ReadU,Unit,ParBuff

header_size=parbuff[2]+parbuff[3]+parbuff[4]+parbuff[5]
header_size1=parbuff[2]
header_size2=parbuff[3]
header_size3=parbuff[4]
header_size4=parbuff[5]

Parbuff=LonArr(Header_size/4)
Point_Lun,Unit,0
ReadU,Unit,Parbuff

DimX=ParBuff[266+(header_size1+header_size2)/4]
DimY=ParBuff[267+(header_size1+header_size2)/4]
DimZ=ParBuff[268+(header_size1+header_size2)/4]

TimePoints=ParBuff[282+(header_size1+header_size2)/4]

OpenR, Unit, Filename,/Get_LUN,Error=Err ; Open the image file

IF Err NE 0 THEN RETURN                    ; Return if file open fails
rawdata=FltArr(2*dimx*dimy)                ; Define array to hold individual
                                           ; slice data
rawdataC=ComplexArr(dimx,dimy,dimz)        ; Define complex array to hold all
                                           ; data
for o=0,dimz-1 do begin                    ; Loop for slices (Z dimension)

DataOffset=(dimx*dimy*4*2)*o+TimePoints*8+HEADER_SIZE  ; Offset for header
                                                       ; and slice number
Point_LUN,Unit,dataoffset                  ; Point to start of slice
READU,Unit,rawdata                         ; Read in slice

RawDataC[*,*,o]=complex(rawdata,0,dimx,dimy)   ; Assemble the slices into a
                                               ; complex array
endfor

Free_LUN,Unit                                  ; Close image file

END
```

Note:
Delphi type SINGLE (data values) corresponds to IDL type FLOAT (4 bytes).
Delphi type DOUBLE (time values) corresponds to IDL type DOUBLE (8 bytes).
Delphi type INTEGER (header values for dimx, dimy and dimz) corresponds to IDL type LONG (4 bytes).

Example: the header is 3584 bytes long. Therefore it will fill an array of LONG 896 in size, and each slice occupies `dimx.dimy.4` (number of bytes per point) `.2` (2 channels) as a result.