

Introduction to pulse programs and experimental setup on Bruker Spectrometers

Dr. Robert Peterson
NMR Facility Manager
UCLA-DOE NMR Core Facility
UCLA Institute for Genomics and Proteomics
and Dept. of Chemistry and Biochemistry

Summary

How to read pulse programs

- basic rules
- definitions
- pulses
- delays
- phases
- translating Bruker pulse programming language into pulse sequence diagrams
- new features in Bruker pulse programming language
- quadrature detection

Tips on setting up NMR experiments

- incremented delays and nd0/nd10
- gradients and gradient selection
- water suppression
- common setup problems
- other helpful information

Bruker Pulse Program Syntax

Basic Rules

Pulses are denoted by p# (0-31).

Delays are denoted by d# (0-31).

All pulses have an associated phase program

The lines of the pulse program are executed sequentially. All the instructions on each line are finished completely before the instructions on the next line are begun.

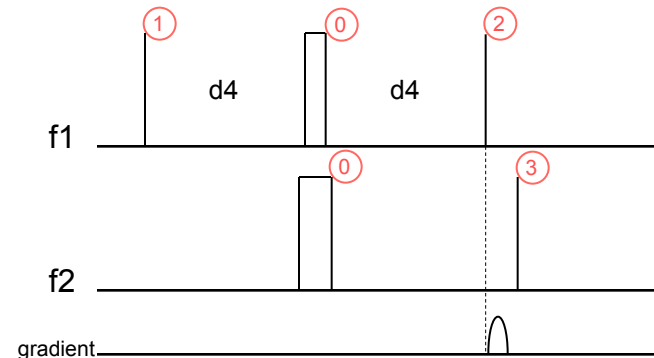
Separate commands on the same line are *begun simultaneously*.

*A semicolon at the beginning of a line indicates a comment.
The compiler ignores these lines.

Example:

“d28=(p4-p2)/2”

```
(p1 ph1):f1  
d4  
(d28 p2 ph0):f1 (p4 ph0):f2  
d4  
(p1 ph2):f1  
d13  
p16:gp1  
d16  
(p3 ph3):f2
```



Definitions

At the top of each pulse program is a list of equations in quotes.

These are simply definitions of some of the parameters used by the pulse program. For example, "p2=p1*2" defines the length of p2 to be twice the length of p1. By setting the value of the parameter p1, the user automatically has set the length of p2. (usually p1 is the ¹H 90° pulse and p2 is the ¹H 180° pulse)

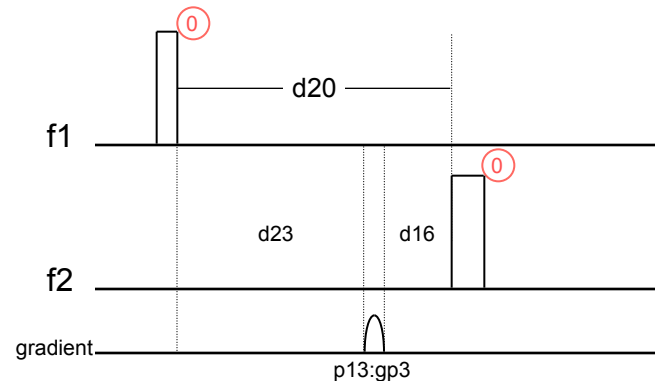
If a definition of a parameter is in the pulse program, you may not set it manually. For example, if p1 is set to 10μsec, p2 will be 20μsec *even if you manually set it to some other value.*

"l3=(td1/2)" sets the loop counter l3 to half the value of td1/2. This allows the user to set the number of increments using eda. If this definition is not present, then you have to set the loop counter manually. For example, if you wanted 512 t1 increments, you would set l3 to 256.

```
"p2=p1*2"  
"p4=p3*2"  
  
"d11=30m"  
"d12=20u"  
"d13=4u"  
"d14=d4-d13-p14"  
"d16=200u"  
"d21=d20-d12"  
"d22=d20"  
"d23=d20-p13-d16"  
"d24=d20-d12"  
  
"d28=p3-p1"  
  
"l3=(td1/2)"
```

Some delays are based on other delays. Usually this is because the pulse sequence requires that two pulses have a particular delay between them. In this short example, the definition of d23 (above) ensures that the pulses p2 and p4 are separated by exactly d20 (which is set by the user). By rearranging the equation, you can see that $d20 = d23 + p13 + d16$.

→ (p2 ph0):f1
d23
p13:gp3*EA
d16
(p4 ph0):f2



Pulses

All pulses are designated p# (0-31). If the pulse is p10, then the parameter p10 is the length of the pulse. The default unit is μsec .

For rectangular pulses, power levels are specified by pl# (0-31). Any pulse number can have any power level number (e.g. p10 could have pl5). The power level of a pulse on a given channel must be set by a pl command, with a delay, on a previous line.

For example:

```
d12 p1:f1
(p3 ph0):f2
d13
p11:gp1
d16
(p1 ph0):f1
```

If no power level is specified in the pulse program, the defaults are pl1 for f1, pl2 for f2, and pl3 for f3. If no channel is specified for a pulse, the default is f1.

In this case, the power level of p1 is pl1.

Shaped pulses are specified like this: (p15:sp5 ph12):f2

For shaped pulses, the length is still specified by p#, but the power level is set by sp#. In this case the length is p15 and the power level is sp5. Shaped pulses also have other parameters:

spoffs5: the offset from the carrier frequency in Hz
spnam5: the name of the shape file

Gradient pulses are specified as: p#:gp# (p11:gp1 in the example above). As with other pulses, p11 is the length. Each gradient has several parameters:

gpnam1: the name of the gradient shape file
gpx1: the X gradient strength
gpy1: the Y gradient strength
gpz1: the Z gradient strength

Delays

Delays are designated d# (0-31). The default unit for delays is seconds.

Fixed length delays can also be specified: e.g. 3u, 10m, etc.

In addition to delays specified by the pulse sequence, delays are needed to allow the hardware time for changing power levels, changing phase, and writing to disk. Delays are also used to allow sample cooling (like during cpd or tocsy sequences) and to allow eddy currents generated by the gradient coils to dissipate.

You should generally use at least 3 μ sec for power level switches or phase changes, although the hardware may not need that long. The hardware on the 800 needs less than 100 nsec to make amplitude or phase changes, but the other instruments (which are older) need longer. Write statements must be behind a delay of at least 10 μ sec, but you should use at least 10msec (the actual time required depends on the computer and the hard drive).

Pulse and delay lengths can be manipulated in the pulse program:

p1*1.5 generates a pulse 1.5 times the length of p1 (often used in cpd or tocsy sequences)

d12*0.5 generates a delay half the length of d12

Phases

Pulse phases are relative phases with respect to the reference phase for signal detection. The phase programs are specified behind the pulse statements, and defined at the end of the pulse program.

Phases are normally specified by 0 (X; 0°), 1 (Y; 90°), 2 (-X; 180°), 3 (-Y; 270°).

For example:

ph3 = 0 0 2 2

At the start of the pulse program, the first phase in each phase program is used. The next phase becomes valid with the next scan or dummy scan.

-NOTE- The phase pointer is not zeroed after the dummy scans. Both ds and ns should be a multiple of the number of phases in the longest phase program.

If the last phase in a phase program is reached, it starts over again. For example, with the following phase programs, and with ns=16:

Phase programs as

written in the pulse program:

ph0 = 0

ph1 = 0 2

ph2 = 1 1 3 3

ph3 = 0 0 0 0 2 2 2 2

ph31=0 2 2 0 2 0 0 2

Phases actually used for all 16 scans:

ph0 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ph1 = 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2

ph2 = 1 1 3 3 1 1 3 3 1 1 3 3 1 1 3 3

ph3 = 0 0 0 0 2 2 2 2 0 0 0 0 2 2 2 2

ph31=0 2 2 0 2 0 0 2 0 2 2 0 2 0 0 2

Translating pulse programs into pulse sequence diagrams

Example 1: SD_ct.ce.hsqhc

```
;CT-CE-HSQC foe 1JCH in labeled samples
;2D H-1/C13 correlation - HSQC
;decoupling in t2
;HC coupling in t1
;carbon high power X channel
;phase sensitive States-TPPI
;7/26/02 LT
```

```
#include <Avance.incl>
```

```
.*****
;
;*****Set in21=in22=in23=in0 *
;*****
```

```
"p2=p1*2"
"p4=p3*2"
```

```
"d11=30m"
"d12=20u"
"d13=4u"
"d14=d4-d13-p14"
"d16=200u"
"d21=d20-d12"
"d22=d20"
"d23=d20-p13-d16"
"d24=d20-d12"
```

```
"d28=p3-p1"
```

```
"l3=(td1/2)"
```

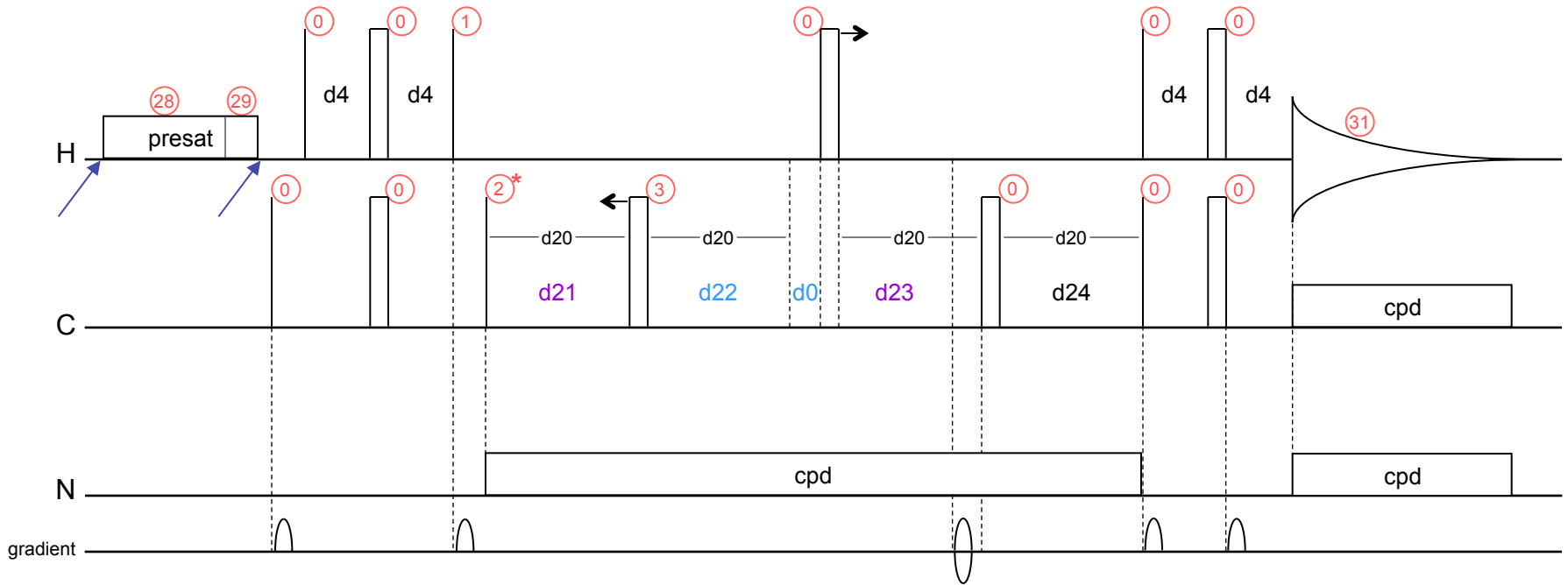
```
1 ze
2 d11*2 do:f2 do:f3
3 d11
4 d11*0.5 pl9:f1
d11*0.5 pl2:f2 pl13:f3
LOCKH_OFF
d12 fq1:f1
(p18*0.8 ph28):f1
d13
(p18*0.2 ph29):f1
d12 fq1:f1
LOCKH_ON
d12 pl1:f1
(p3 ph0):f2
d13
p11:gp1
d16
(p1 ph0):f1
```

```
d4
(d28 p2 ph0):f1 (p4 ph0):f2
d4
(p1 ph1):f1
d13
p12:gp2
d16
(p3 ph2):f2
d12 cpd3:f3
d21 ; decrementation
(p4 ph3):f2
d22 ; incrementation
d0 ; incrementation
(p2 ph0):f1
d23 ; decrement
p13:gp3*EA ; switching the sign
d16
(p4 ph0):f2
d24
d12 do:f3
(p3 ph0):f2 (p1 ph0):f1
d13
p14:gp4
d14
(d28 p2 ph0):f1 (p4 ph0):f2
d13
p14:gp5
d14 pl12:f2
go=2 ph31 cpd2:f2 cpd3:f3
d11 do:f2 do:f3 wr #0 if #0 ip2 zd
d11*0.5 ip2
d11*0.5 igrad EA
lo to 3 times 2
d11*0.25 id0
d11*0.25 id22
d11*0.25 dd21
d11*0.25 dd23
lo to 4 times l3
LOCKH_OFF
exit

ph0 = 0
ph1 = 1 3
ph2 = 0 0 2 2
ph3 = 0 0 0 1 1 1 1
ph28 = 0
ph29 = 1
ph31= 0 2 2 0 2 0 2
```

```
;pl1: H-1 power level
;p12: C-13 power level
;p19: H-1 power level for water presat
;p12:C-13 power level for C-13 decoupling
;p1 : 90 degree pulse
;p2 : 180 degree pulse
;p3 : 90 degree decoupler pulse
;p4 : 180 degree decoupler pulse
;pcpd2: decoupler pulse for GARP decoupling
;d4 : 1/4J(HC) (~1.25-1.6 ms)
;d0 : (in0-p2-p3*4/pi)/2
;p16: gradient pulse (600 us)
;d11: I/O delay for disc (30 ms)
;d12: delay for power switching
;d13: delay compensation 4us
;in0: 1/2SW(X)
;nd0: 2
;NS: *16, minimum 2
;td1: # of t1-experiments
;MC2: States-TPPI
```



SD_ct.ce.hsqchc



d0 and d22: incremented delays

d21 and d23: decremented delays

phase 2: cycled for States-TPPI quadrature detection

 = location of frequency switches

Newer features in Bruker pulse programming language

Alignment of parallel pulse trains:

Older method of centering pulses

“ $d28=(p4-p2)/2$ ”

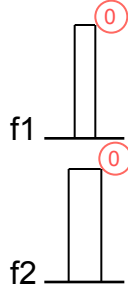
(d28 p2 ph0):f1 (p4 ph0):f2

Newer method of centering pulses

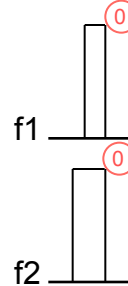
(center (p2 ph0):f1 (p4 ph0):f2)

New alignment commands

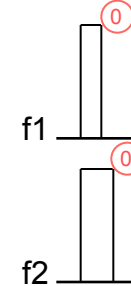
(center (p2 ph0):f1 (p4 ph0):f2)



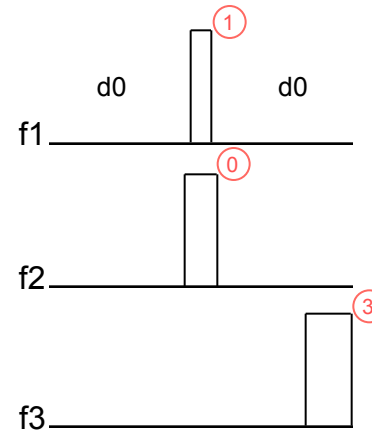
(ralign (p2 ph0):f1 (p4 ph0):f2)



(lalign (p2 ph0):f1 (p4 ph0):f2)



(
 refalign (d0 p2 ph1 d0):f1
 center (p4 ph0):f2
 ralign (p22 ph3):f3
)



Newer features in Bruker pulse programming language

Comparison operator “larger”

```
"DELTA3=d0*2+larger(p14,p22)-p14"
```

This allows the delay DELTA3 to be calculated correctly, regardless of whether p14 or p22 is larger.

A clever way to set offset frequencies of shaped pulses

```
"spoff5=bf2*(cnst21/1000000)-o2"
```

In this example, o2 is the offset frequency of the carbon channel, which would be set near 40ppm for protein experiments. Shaped pulses using sp5 are used to hit the carbonyls, and so require an offset frequency (spoffs5). If this definition is present, all you have to do is set cnst21 to 175 (the frequency, in ppm, of the carbonyls).

Note that this method allows you to specify the offset frequency in ppm, making the whole setup independent of field strength.

Newer features in Bruker pulse programming language

Conditional execution of labeled pulse program commands

The “ifdef” and “else” statements are used to select sections of the pulse program for conditional execution. In the past, in order to change which section of the program was active, the user had to comment or uncomment a “define” statement.

This often leads to permission problems.

```
 ;#define REFOCUS
 ...
 #ifdef REFOCUS
 (p2 ph0):f1
 DELTA
 #endif
```

The new feature regarding conditionals is that the user can define conditionals on the command line together with the “zg” command. For example:

<pre># ifdef LABEL_CN “DELTA=p16+d16+larger(p2,p22)+d0*2 # else “DELTA=p16+d16+p2+d0*2” # endif /*LABEL_CN*/</pre>	. . .	<pre># ifdef LABEL_CN (center (p2 ph5) (p22 ph1):f3) # else (p2 ph5) # endif /*LABEL_CN*/</pre>
--	-------	--

If the sample is nitrogen labeled, then defining LABEL_CN will result in decoupling nitrogen, as well as proton, during the t_1 time. To define it, the user can type “zg -DLABEL_CN when starting the experiment. Alternatively, -DLABEL_CN can be stored under ZG_OPTNS, and it will be defined when “zg” is entered (you can set it in ASED or in EDA).

Newer features in Bruker pulse programming language

Method of quadrature detection

In the old style programming, the looping for quadrature detection was written explicitly. This has the advantage that you can see in the pulse program exactly how the loops work. On the right is an example of quadrature looping for a 3D experiment written in the older style.

```
go=2 ph31 cpd3:f3
d11 do:f3 wr #0 if #0 zd
d11 ip5*2 igrad EA
lo to 3 times 2
d11 id10
d11 d30
lo to 4 time l13
d11 rd10 ip3
d11 rd30
lo to 5 times 2
d11 id0
d11 id20
d11 dd29
lo to 6 times 13
```

Below is the way the same loop structure is written in the new style:

```
go=2 ph31 cpd3:f3
d11 do:f3 mc #0 to 2
  F1PH(rd10 & rd30 & ip3, id0 & id20 & dd29)
  F2EA(igrad EA & ip5*2, id10 & dd30)
```

The advantage of the new method is its flexibility. In the new method, the parameter “FnMODE” must be set. This allows the same pulse program to have different methods of quadrature detection. In the above example, FnMODE can be States-TPPI, States, TPPI, &etc in F1, but must be echo-antiecho in F2 (any time “EA” appears in the parentheses, FnMODE must be echo-antiecho). You should be familiar with both styles, because you will undoubtedly encounter both in your work.

BE CAREFUL! If the old style looping is used, FnMODE must be set to “undefined”.

How does the looping for quadrature detection actually work?

States-TPPI looping

"l1=td1/2"

2 d11*2

3 d11*2

4 d12 do:f3

.

.

.

go=2 ph31 cpd3:f3

d11 do:f3 wr #0 if #0 zd

d11 ip1

lo to 3 times 2

d11 id0

d11 (ip31*2)

lo to 4 times l1

30m BLKGRAD

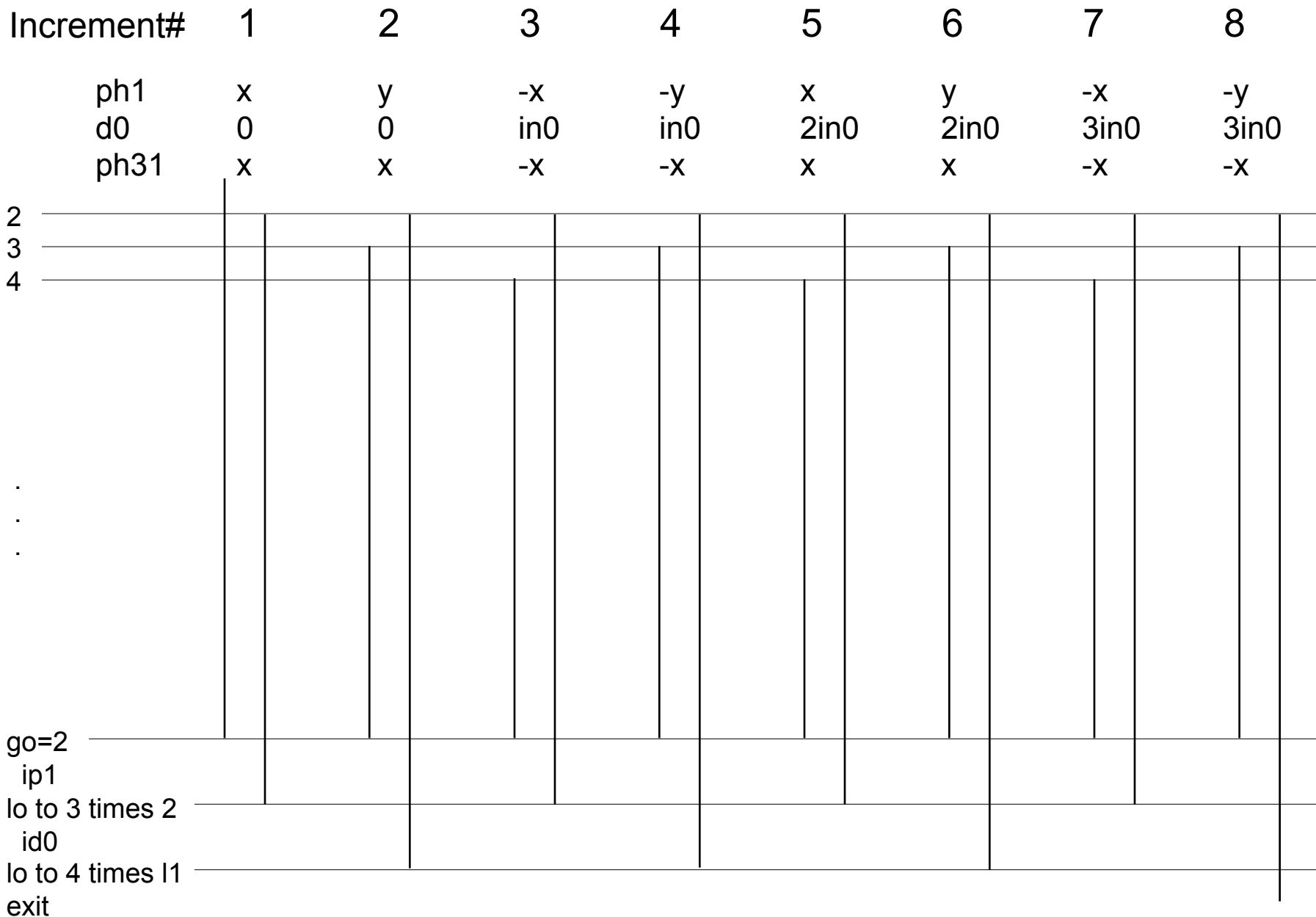
exit

<u>Increment</u>	<u>ϕ_1</u>	<u>t_1 value</u>	<u>receiver</u>
1	x	0	x
2	y	0	x
3	-x	2Δ	-x
4	-y	2Δ	-x

The part following acquisition consists of 3 nested loops

- 1) go=2
loops times (ns-1)
- 2) lo to 3 times 2
loops back to 3 one time
- 3) lo to 4 times l1
loops to 4 (l1-1) times.

The definition at the top defines l1 to be 1/2 of td1 (TD in F1 that can be entered in eda). If the pulse sequence does not contain this definition you must set l1 manually.



This schematic shows the complete looping of a States-TPPI-style 2D pulse sequence with $ns=2$ and $I1=4$ ($TD1=8$).

Translating pulse programs into pulse sequence diagrams

Example 2: hncacbgp3d (Bruker standard hncacb)

```
;hncacbgp3d
;avance-version (02/05/31)
;HNCACB
;3D sequence with
; inverse correlation for triple resonance using multiple
; inept transfer steps
;
; F1(H) -> F3(N) -> F2(Ca -> Cb,t1) -> F3(N,t2) -> F1(H,t3)
;
;on/off resonance Ca and C=O pulses using shaped pulse
;phase sensitive (t1)
;phase sensitive using Echo/Antiecho gradient selection (t2)
;using constant time in t2
;(use parameterset HNCACBGP3D)
;
;M. Wittekind & L. Mueller, J. Magn. Reson. B 101, 201-205 (1993)
;D.R. Muhandiram & L.E. Kay, J. Magn. Reson. B 103, 203-216 (1994)
```

```
prosol relations=<triple>
```

```
#include <Avance.incl>
#include <Grad.incl>
#include <Delay.incl>
```

```
"d0=3u"
```

```
"d11=30m"
```

```
"d13=4u"
```

```
"d21=5.5m"
```

```
"d23=12.4m"
```

```
"d26=2.3m"
```

```
"d28=3.6m"
```

```
"in29=in10"
```

```
"in30=in10"
```

```
"d10=d23/2-p14/2"
```

```
"d29=d23/2-p14/2-p26-d21-4u"
```

```
"d30=d23/2-p14/2"
```

```
"DELTA1=p16+d16+d13+4u"
"DELTA2=d23-d21-p26"
"DELTA3=d0*2+larger(p14,p22)-p14"
"DELTA4=d21-p16-d16-4u"
```

```
"spoff2=0"
"spoff3=0"
"spoff5=bf2*(cnst21/1000000)-o2"
"spoff8=0"
```

```
aqseq 321
```

```
1 ze
  d11 pl16:f3
2 d11 do:f3
3 d1 pl1:f1
  (p1 ph1)
  d26 pl3:f3
  (center (p2 ph1) (p22 ph1):f3 )
  d26 UNBLKGRAD
  (p1 ph2):f1
```

```
4u pl0:f1
(p11:sp1 ph1:r):f1
4u
p16:gp1
d16
```

```
(p21 ph3):f3
d21 pl19:f1
(p26 ph2):f1
DELTA2 cpds1:f1 ph1
(center (p14:sp3 ph1):f2 (p22 ph1):f3 )
d23
(p21 ph1):f3
```

```
(p13:sp2 ph4):f2
d28
(p14:sp3 ph1):f2
d28
(p13:sp8 ph2):f2
```

```
d0
(center (p14:sp5 ph1):f2 (p22 ph8):f3 )
d0
4u
(p14:sp3 ph1):f2
DELTA3
(p14:sp5 ph1):f2
4u
```

```
(p13:sp2 ph9):f2
d28
(p14:sp3 ph1):f2
d28
(p13:sp8 ph10):f2
```

```
(p21 ph1):f3
d30
(p14:sp5 ph1):f2
d30
(center (p14:sp3 ph1):f2 (p22 ph8):f3 )
d10
(p14:sp5 ph1):f2
d29
4u do:f1
(p26 ph7):f1
4u
p16:gp2*EA
d16
DELTA4 pl1:f1
```

```
(center (p1 ph1) (p21 ph5):f3 )
d26
(center (p2 ph1) (p22 ph1):f3 )
d26
(center (p1 ph2) (p21 ph6):f3 )
d26
```

```
(center (p2 ph1) (p22 ph1):f3 )
d26
(p1 ph1)
DELTA1
(p2 ph1)
d13
p16:gp3
d16 pl16:f3
4u BLKGRAD
go=2 ph31 cpd3:f3
d11 do:f3 mc #0 to 2
```

```
F1PH(rd10 & rd29 & rd30 & ip9 & ip10, id0 & dp9*2)
F2EA(igrad EA & ip6*2, id10 & id29 & dd30)
```

```
exit
```



```

ph1=0
ph2=1
ph3=0 0 0 0 0 0 2 2 2 2 2 2 2 2
ph4=0
ph5=0 0 2 2
ph6=3 3 1 1
ph7=3
ph8=0 0 0 0 2 2 2 2
ph9=3 1
ph10=0 2
ph31=0 2 2 0 0 2 2 0 2 0 0 2 2 0 0 2

```

```

;p10 : 120dB
;p1 : f1 channel - power level for pulse (default)
;p3 : f3 channel - power level for pulse (default)
;p16: f3 channel - power level for CPD/BB decoupling
;p19: f1 channel - power level for CPD/BB decoupling
;sp1: f1 channel - shaped pulse 90 degree (H2O on resonance)
;sp2: f2 channel - shaped pulse 90 degree (Cali on resonance)
;sp3: f2 channel - shaped pulse 180 degree (Cali on resonance)
;sp5: f2 channel - shaped pulse 180 degree (C=O off resonance)
;sp8: f2 channel - shaped pulse 90 degree (Cali on resonance)
;
;      for time reversed pulse
;p1 : f1 channel - 90 degree high power pulse
;p2 : f1 channel - 180 degree high power pulse
;p11: f1 channel - 90 degree shaped pulse      [2 msec]
;p13: f2 channel - 90 degree shaped pulse
;p14: f2 channel - 180 degree shaped pulse
;p16: homospoil/gradient pulse                [1 msec]
;p21: f3 channel - 90 degree high power pulse
;p22: f3 channel - 180 degree high power pulse
;p26: f1 channel - 90 degree pulse at p19
;d0 : incremented delay (F1 in 3D)            [3 usec]
;d1 : relaxation delay; 1-5 * T1
;d10: incremented delay (F2 in 3D) = d23/2-p14/2
;d11: delay for disk I/O                      [30 msec]
;d13: short delay                            [4 usec]
;d16: delay for homospoil/gradient recovery
;d21: 1/(2J(NH))                             [5.5 msec]

```

```

;d23: 1/(4J(NCa))                            [12.4 msec]
;d26: 1/(4J'(NH))                            [2.3 msec]
;d28: 1/(4J(CaCb))                           [3.6 msec]
;d29: incremented delay (F2 in 3D) = d23/2-p14/2-p26-d21-4u
;d30: decremented delay (F2 in 3D) = d23/2-p14/2
;cnst21: CO chemical shift (offset, in ppm)
;cnst23: Caliphatic chemical shift (offset, in ppm)
;o2p: Caliphatic chemical shift (cnst23)
;in0: 1/(2 * SW(Cali)) = DW(Cali)
;nd0: 2
;in10: 1/(4 * SW(N)) = (1/2) DW(N)
;nd10: 4
;in29: = in10
;in30: = in10
;NS: 8 * n
;DS: >= 16
;td1: number of experiments in F1
;td2: number of experiments in F2      td2 max = 2 * d30 / in30
;FnMODE: States-TPPI (or TPPI) in F1
;FnMODE: echo-antiecho in F2
;cpds1: decoupling according to sequence defined by cpdprg1
;cpd3: decoupling according to sequence defined by cpdprg3
;pcpd1: f1 channel - 90 degree pulse for decoupling sequence
;pcpd3: f3 channel - 90 degree pulse for decoupling sequence

;use gradient ratio:  gp 1 : gp 2 : gp 3
;                      30 : 80 : 8.1

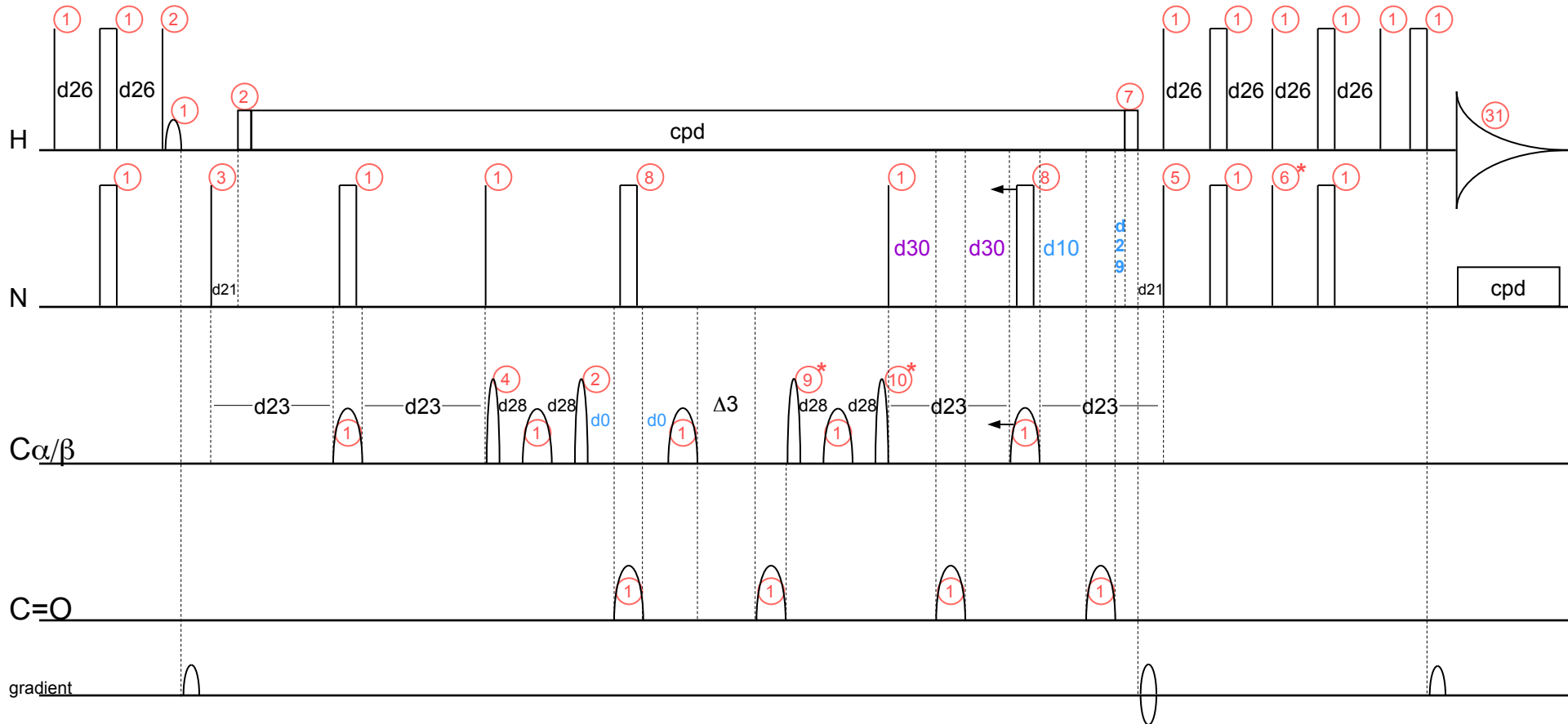
;for z-only gradients:
;gpz1: 30%
;gpz2: 80%
;gpz3: 8.1%

;use gradient files:
;gpnam1: SINE.100
;gpnam2: SINE.100
;gpnam3: SINE.100

```

;\$ld: hncacbgp3d,v 1.15 2002/06/12 09:04:44 ber Exp \$

hncacb3d



d0, d10, and d29: incremented delays

d30: decremented delay

phases 6, 9, and 10: cycled for States-TPPI quadrature detection

Tips On Setting Up NMR Experiments

Tip #1: Look at your pulse sequence!

Incremented delays

If any of the indirect dimensions is constant time, you have to make sure to set all the *in* values for that dimension the same. Usually there are comments telling you which ones have to be equal. Or you can look at the quadrature loops to see which ones go together.

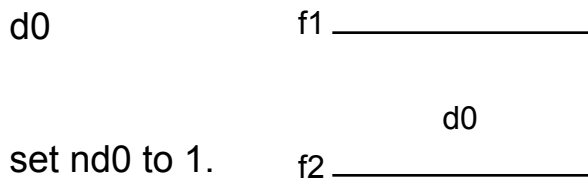
If you set the sweep widths with the *sw* parameter in *eda*, that will automatically set *in0* (for f_1) or *in10* (for f_2). You then have to set the other *in* values to match.

$$*In\ xwinmr, SWH = \frac{1}{IN0*ND0}$$

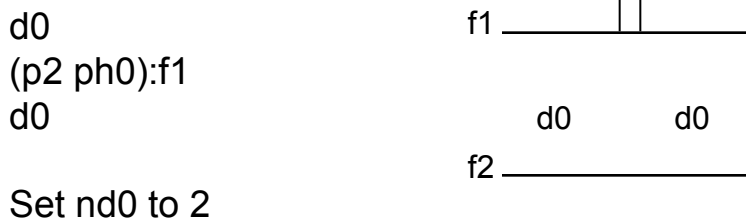
nd0

If you are using States-TPPI or echo-antiecho quadrature detection (almost all experiments do), then *nd0* is simply the number of times *d0* appears in the pulse sequence.

For the simplest evolution:

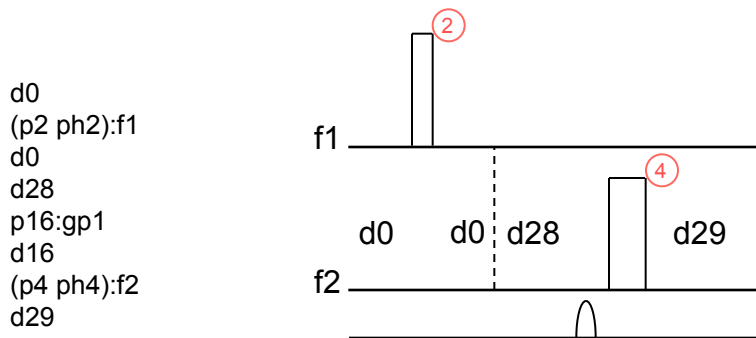


If the evolution has a 180° decoupling pulse:



nd0

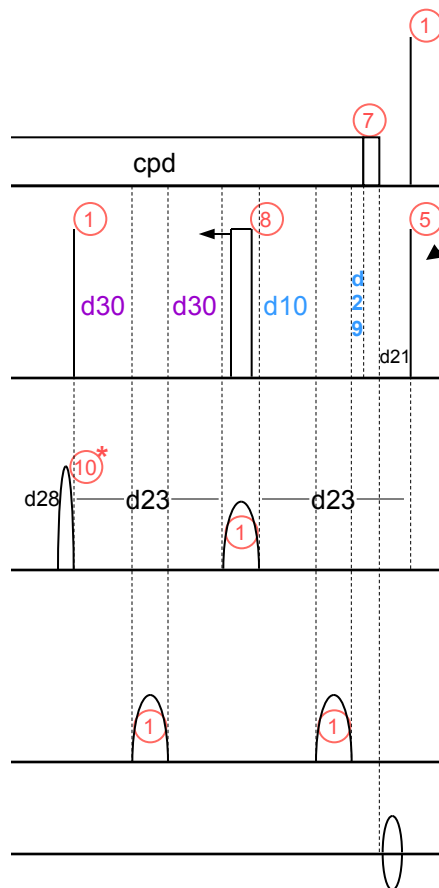
If the evolution is constant time, it's a little more complicated. In general, if the evolution looks like this:



with in0, dd28, and dd29 in the quadrature looping, nd0 should be set to 2. And of course, in28 and in29 should be set equal to in0.

If the evolution looks like this:

d30
(p14:sp5 ph1):f2
d30
(center (p14:sp3 ph1):f2 (p22 ph8):f3)
d10
(p14:sp5 ph1):f2
d29
4u do:f1
(p26 ph7):f1
4u
p16:gp2*EA
d16
DELTA4 pl1:f1



-note-
This is a combined chemical shift evolution, and antiphase
-> in-phase evolution for the
C -> N INEPT transfer

with dd30, id10, and id29 in the quadrature looping, nd10 should be set to 4 (and in30 and in29 set equal to in10).

Gradient Selection

How can I tell if my experiment uses gradient selection?

Look into the pulse program and search for *EA*. If *EA* appears in the body of the pulse program, then gradient selection is used.

If your pulse program does use gradient selection, then the selection gradients must be in the right ratio (4:1 for ^{13}C , 10:1 for ^{15}N). In Bruker sequences, the selection gradients will normally be the one multiplied by *EA*, and the final gradient. It can occasionally be tricky – for example in the experiment *SD_ct.ce.hsqchc*, shown above, the selection gradients are the last *three* gradients. The first of these (multiplied by *EA*) dephases the magnetization while it's on carbon, and the last *two* rephase the magnetization on proton. Therefore the gradient ratios must be 8:1:-1.

In Bruker standard pulse programs, the correct ratios should be in the comments.

But for purposes other than coherence selection, the exact length and strength of gradients is not so important.

If the gradient appears inside a fixed-length delay, you have to make sure the gradient will fit inside the delay.

The cryoprobes and the QXI probe only have Z gradients.

The recovery delay after gradients should be at least 100 μsec .

***Gradient shapes need more than 5 μsec per point in order to run properly. So, for example, a gradient with shape *sine.100* (a shape file with 100 points) needs to be *longer than 500 μsec* .**

Water Suppression

Presaturation:

The length of the presat (p18 or d1) should be the same as you would use for d1 (1-2 seconds). The power (usually pl9) should be around 50dB for water, 70dB for D₂O.

Water-Flip-Back and WATERGATE:

For setup purposes, you only have to adjust the selective 90° pulses. In the pulse program, these pulses will either look something like this:

```
(p11 ph11:r):f1
```

or like this:

```
(p11:sp1 ph11:r):f1
```

They are selective 90° pulses on water, so of course they will always be on channel 1.

To optimize the WFB WATERGATE water suppression, you should start the experiment in *gs* mode. In this mode you can interactively optimize the relevant parameters. The parameters you should optimize are:

o1

power levels: each power level associated with one of the selective 90s
(the pulse length should be at least 1msec)

phcor: Any time the phase program has a :r behind it, the phase can be changed by small amounts. Usually the selective 90s on water are written this way, and they should all be optimized in *gs* mode. In the above example, phcor11 should be optimized. (I often look for selective 90s on water in the pulse program by searching for “:r”)

Common Setup Problems

If your experiment has a constant time evolution, you can only have as many increments in that dimension as will fit. For example, in the hncacbgp3d experiment above, the delay d30 is decremented in the t_2 dimension. It is defined as follows: “d30=d23/2-p14/2“, and d23 is defined to be 12.4msec. So d30 will have a defined length (say 12.2msec) that can't be changed by the user. If the nitrogen sweep width is 2050 Hz, it would only be possible to have 202 increments in t_2 .

$$1/(4*SWHZ)=DW \sim 122\mu\text{sec} \text{ and } 100*122\mu\text{sec}=12.2\text{msec}$$

(d30 is decremented only every other increment - so with 202 total t_2 values, d30 will be decremented 100 times)

If you had more than 202 increments, d30 would become negative, which is of course not possible. So if you set td2 to more than 202 and type *zg* or *expt*, you will get an error that d30 becomes negative by too many dd30 commands. In this case the only things you can do are to increase the sweep width in that dimension, or to reduce the number of increments.

There's another problem that gives the same type of error. If the incremented delays d0 or d10 are defined in the pulse program by a formula,

$$“d0=(in0/2)-(p5*2/3.14)-p1”$$

*This is done so that the phases in f_1 are known in advance.
In this case $ph0=90$ and $ph1=-180$.

you will sometimes get an error that d0 becomes negative by too many dd0 commands. In this case the error is a red herring because there are no dd0 commands. The reason you get this error is that $(in0/2)-(p5*2/3.14)-p1$ is negative (or because $p5*2/3.14+p1$ is larger than $in0/2$). The easiest way around this is to decrease p5 or p1 by going to higher power. If that isn't possible, you can redefine d0:

(next page)

“d0=in0-(p5*2/3.14)-p1”

In this case, ph0 will be 180 and ph1 will be -360.

or

“d0=(in0*3/2)-(p5*2/3.14)-p1”

In this case, ph0 will be 270 and ph1 will be -540.

Or you can define d0 like this:

“d0=3u”

And then you will have to manually phase the indirect dimension.

Of course, you can't redefine d0 unless you have write permission, which you probably don't. In that case, the only thing you can do is to save the (new) pulse program under a new name, and change the parameter PULPROG to the new name.

Most important!

Check the power levels for all long pulses or pulse trains! Examples are presaturation pulses, TOCSY pulse trains, ROESY mixing pulses, and cpd decoupling pulse trains.

Presaturation pulses are either written using the cw command, or as a conventional pulse:

In this version, the length of the presaturation is d1, and the power is pl9:

```
d12 pl9:f1
d1 cw:f1 ph29
4u do:f1
d12 pl1:f1
```

In this version, the length is p18, and the power is pl9:

```
d12 pl9:f1
p18 ph29
d12 pl1:f1
```

Presaturation pulses are usually long (1-2 sec), so the power has to be relatively weak. The presaturation power shouldn't be higher than 50dB (*i.e.* 45dB is too high).

Good values to use are 50dB for H₂O samples, and 70dB for D₂O samples.

If your experiment has a TOCSY pulse train, you need to identify it in the pulse sequence. It will be a long series of pulses, often multiplied by strange numbers, and with a looping statement at the end.

Here is the beginning of the dipsi2 TOCSY sequence from the experiment “dipsi2etgp”:

```
...  
p1 ph3  
d20 pl10:f1  
  
;begin DIPSII2  
  
4 p6*3.556 ph23  
  p6*4.556 ph25  
  p6*3.222 ph23  
  p6*3.167 ph25  
  p6*0.333 ph23  
  p6*2.722 ph25  
  p6*4.167 ph23  
  p6*2.944 ph25  
  p6*4.111 ph23  
  
...  
lo to 4 times l1
```

At the top of the pulse sequence, there is a definition that fixes the overall length of the dipsi2 to d9:

```
“FACTOR1=(d9/(p6*115.112))/2+0.5”  
“L1=FACTOR1*2”
```

The important things here are the pulse length (p6), the power level (pl10), and the overall length of the dipsi2 sequence (d9). P6 should normally not be less than 25μsec, and pl10 should be calibrated for a 90° pulse of length p6.

If your experiment is a ROESY, there is one very long pulse which is the ROESY spin-lock. Here's part of a simple ROESY pulse sequence:

```
d12 pl9:f1  
d1 cw:f1 ph29  
d13 do:f1  
d12 pl1:f1  
p1 ph1  
d0  
4u pl11:f1  
p15 ph2  
go=2 ph31
```

The ROESY spin-lock is p15, and the power level is pl11. Usually for experimental purposes, the spin-lock should be 200msec or less. But you have to be very careful not to apply too much power. For the cryoprobes, the field strength of the spin-lock pulse can't be higher than 2.5 KHz - or the power corresponding to a 100 μ sec 90° pulse.

Of the various ways you can overpower the probe, probably the most dangerous is composite pulse decoupling (cpd). The reason cpd is so dangerous is that you can only set the length of the decoupling in a somewhat roundabout way. This kind of decoupling runs during the entire acquisition, so the only way to change the length of the decoupling is to change the length of the acquisition.

Here are the last few lines of a typical pulse program that uses cpd decoupling on carbon and nitrogen during the acquisition:

```
d12                                     length of decoupling is aq
p16:gp2
d17 pl12:f2 pl13:f3
d12 BLKGRAD
go=2 ph31 cpd2:f2 cpd3:f3
d11 do:f2 do:f3 wr #0 if #0 zd
```

Here the power level for cpd2 is pl12 and the power level for cpd3 is pl13. The lengths of the decoupling pulses are pcpd2 and pcpd3. Typical values are pcpd2=70 μ sec (^{13}C), and pcpd3=170 μ sec (^{15}N), with corresponding power levels (pcpd2 and pcpd3 are 90° pulses). To find the current acquisition time, you can type “aq”. But to change the acquisition time, you can only increase the sweep width (of the directly detected dimension), or decrease TD (the number of points acquired in the directly detected dimension).

***On cryoprobes, acquisition is not allowed to be longer than 140msec, if decoupling is being used.** And if there is decoupling on both channels (as above), you must reduce the power. You can use pcpd2=100 μ sec, and pcpd3=200 μ sec, with corresponding power levels.