Electronic Supplementary Material (ESI) for Polymer Chemistry.
This journal is © The Royal Society of Chemistry 2015

Synthesis and Microstructural Characterization of Poly(chlorotrifluoroethylene-*co*-vinylidene chloride) Copolymers

Gérald Lopez,[†] *Chun Gao,*[‡] *Linlin Li,* [‡] *Faith J. Wyzgoski,*[ǯ] *Alagu Thenappan,*[§]
Peter L. Rinaldi,[†,Π]* *and Bruno Ameduri*[†]*

[†]Ingénierie & Architectures Macromoléculaires, Institut Charles Gerhardt, École Nationale Supérieure de Chimie de Montpellier, 8 Rue de l'École Normale, 34296 Montpellier, France. [‡]Department of Chemistry, University of Akron, 190 East Buchtel Commons, Akron, OH 44325-3601, USA. [ǯ]Department of Chemistry and Biochemistry, The Ohio State University, 1760 University Drive, Mansfield, Ohio 44906. [§]Honeywell, 101 Columbia Road, Morristown, N.J., 07962, USA. [Π]College of Chemistry, Chemical Engineering and Materials Science, Soochow University, 199 Renai Road, Suzhou Industrial Park, Suzhou, Jiangsu Province, 21513, P. R. China.

Corresponding Authors

*Prof. P. L. Rinaldi: Tel: +1330-972-5990; E-Mail: peter.rinaldi@uakron.edu

*Dr B. Ameduri: Tel: +33-467-144-368; E-Mail: bruno.ameduri@enscm.fr

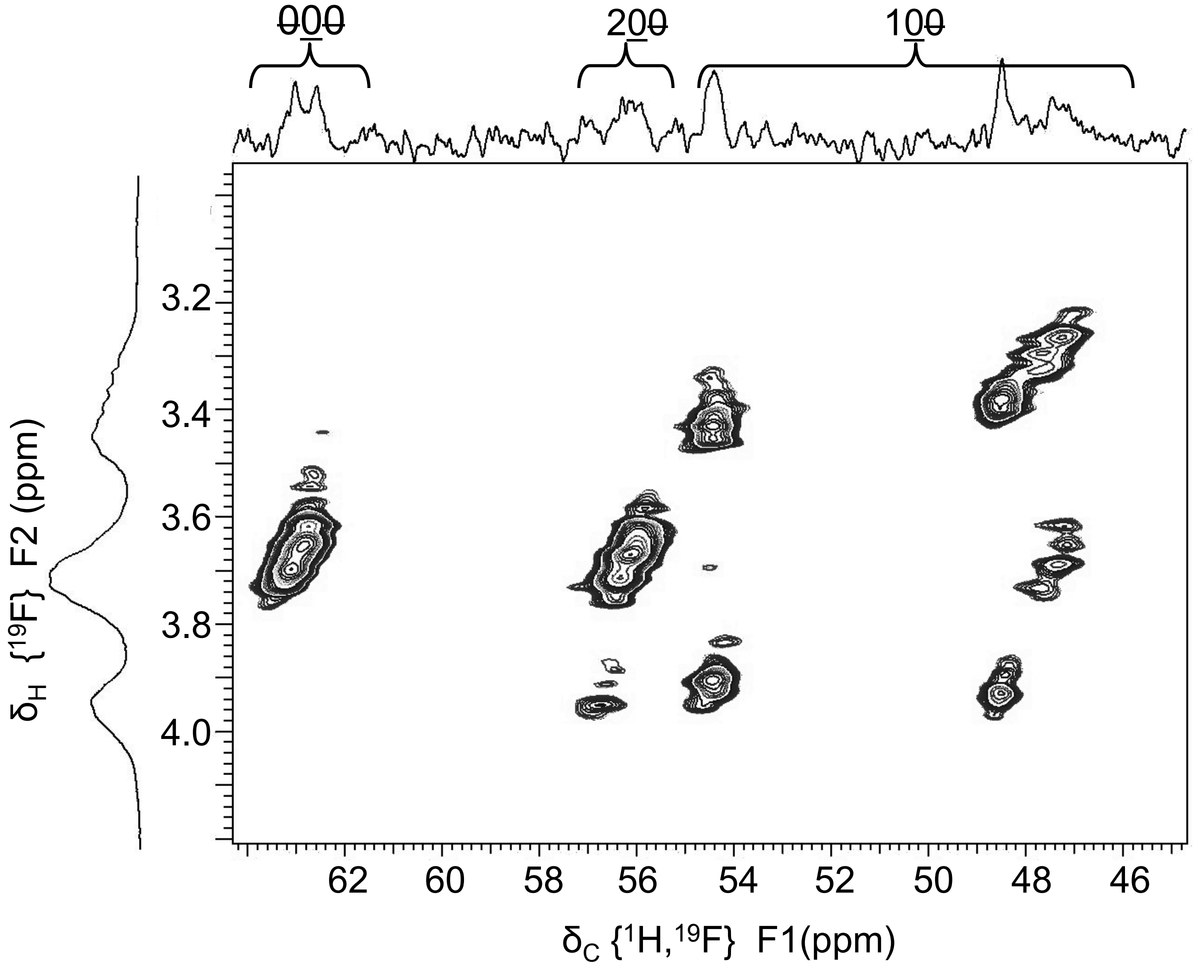

Figure S1. Selected region from the 500 MHz ^{1}H{^{13}C} gHSQC spectrum of poly(CTFE-*co*-VDC) obtained in toluene-d$_{8}$ at 90 °C.

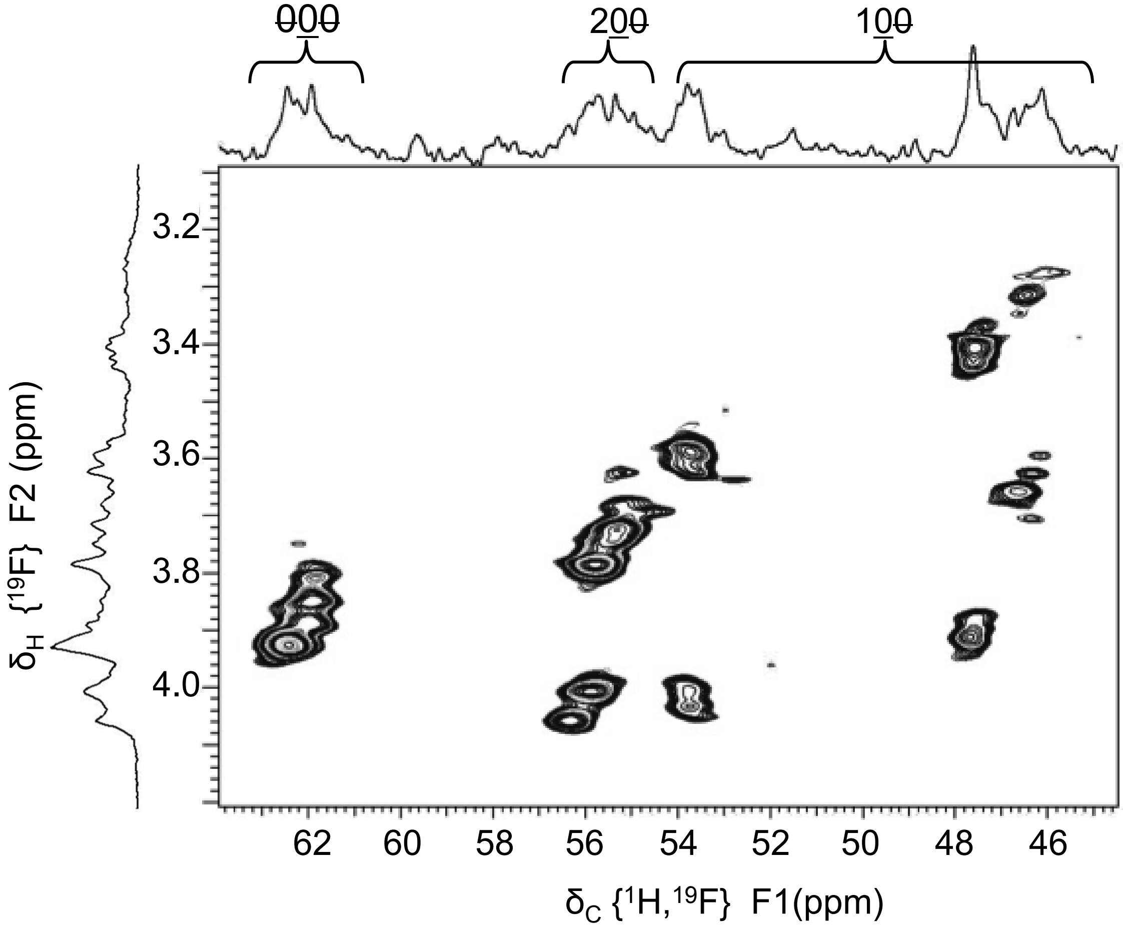

Figure S2. Selected region from the 500 MHz ^{1}H{^{13}C} gHSQC spectrum of poly(CTFE-*co*-VDC) obtained in $CDCl_3$ at 50 °C.

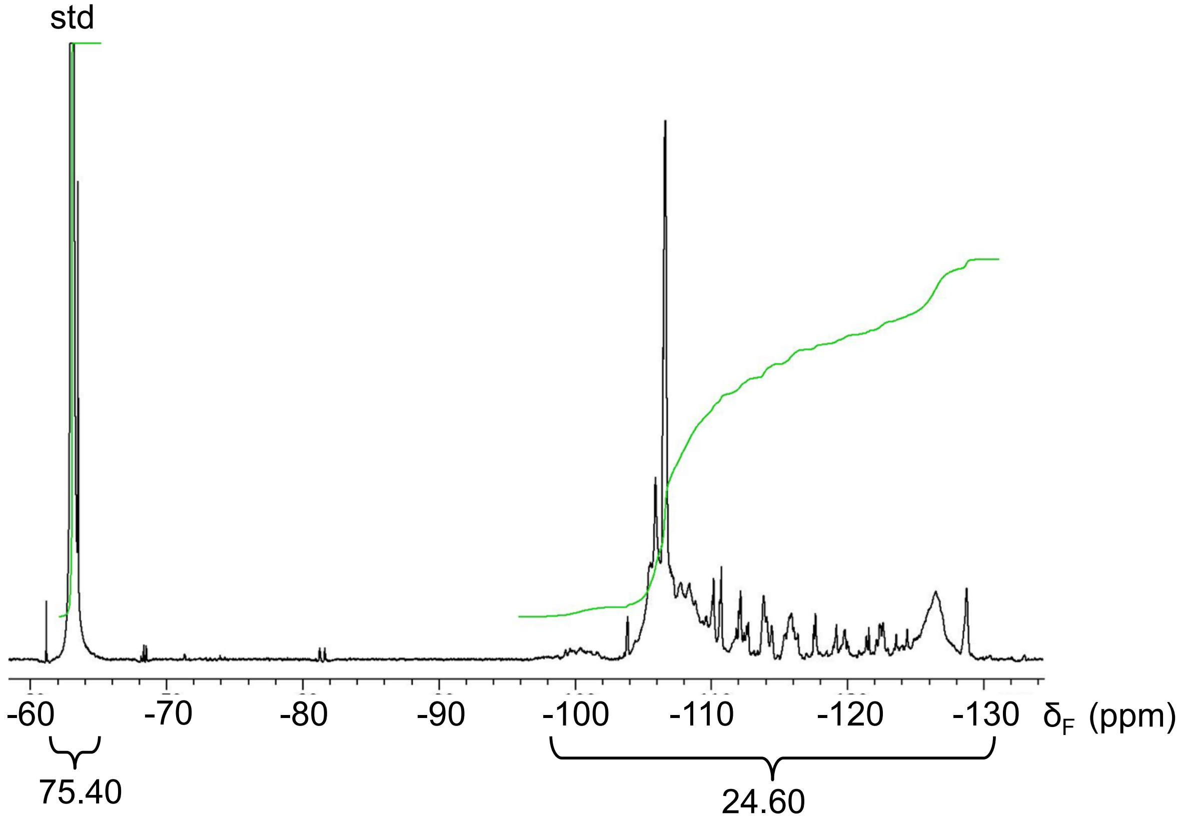

Figure S3. Quantitative 470 MHz $^{19}F\{^1H\}$ 1D-NMR spectrum of poly(CTFE-*co*-VDC) with internal standard in $CDCl_3$ at 50 ^0C.

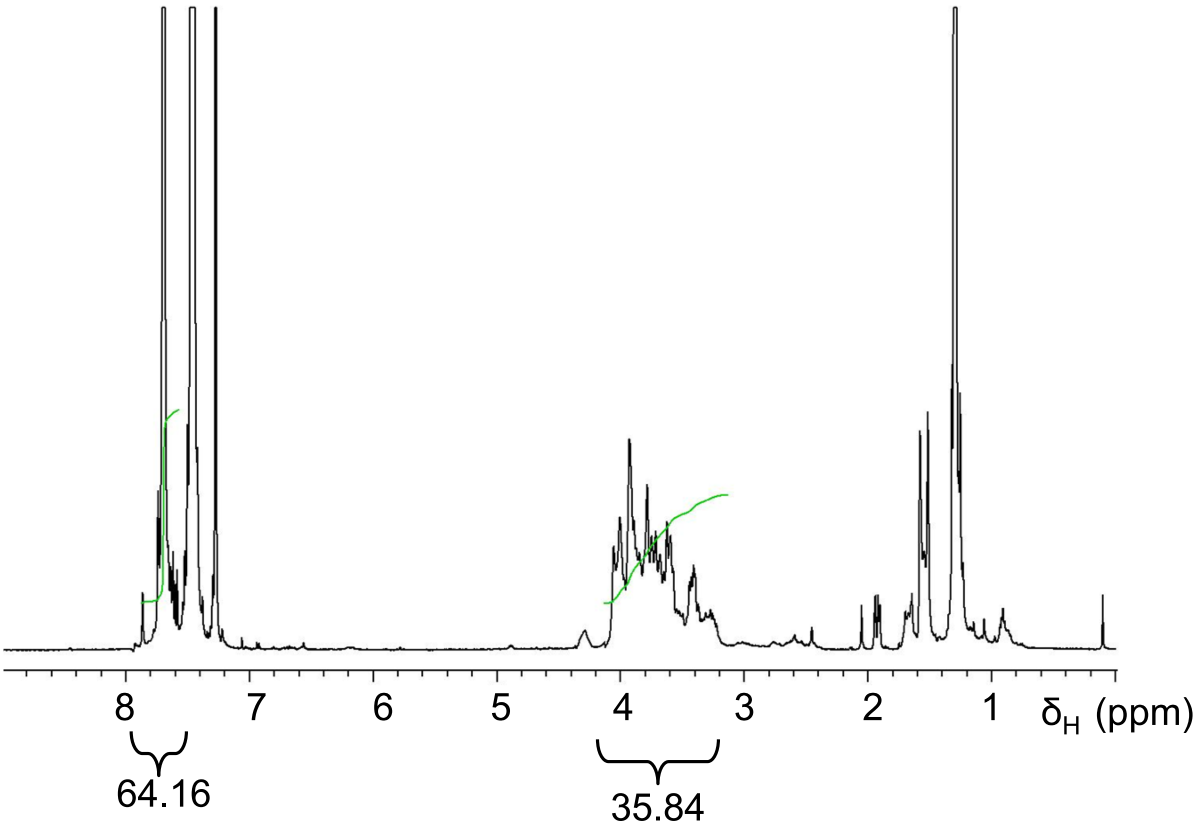

Figure S4. Quantitative 500 MHz $^{1}H\{^{19}F\}$ 1D-NMR spectrum of poly(CTFE-*co*-VDC) with internal standard in $CDCl_3$ at 50 ^{0}C.

Determination of P_{VDC} from Fig. S3 and Fig. S4:

$$\frac{N_{VDC}}{N_{std}} = \frac{A_{VDC}\,/\,2}{A_{std}\,/\,1} = \frac{35.84\,/\,2}{64.16\,/\,1} = 0.28$$

$$\frac{N_{CTFE}}{N_{std}} = \frac{A_{CTFE}\,/\,3}{A_{std}\,/\,3} = \frac{24.60\,/\,3}{75.40\,/\,3} = 0.33$$

$$P_{VDC} = \frac{N_{VDC}}{N_{VDC} + N_{CTFE}} * 100\% = 46\%$$

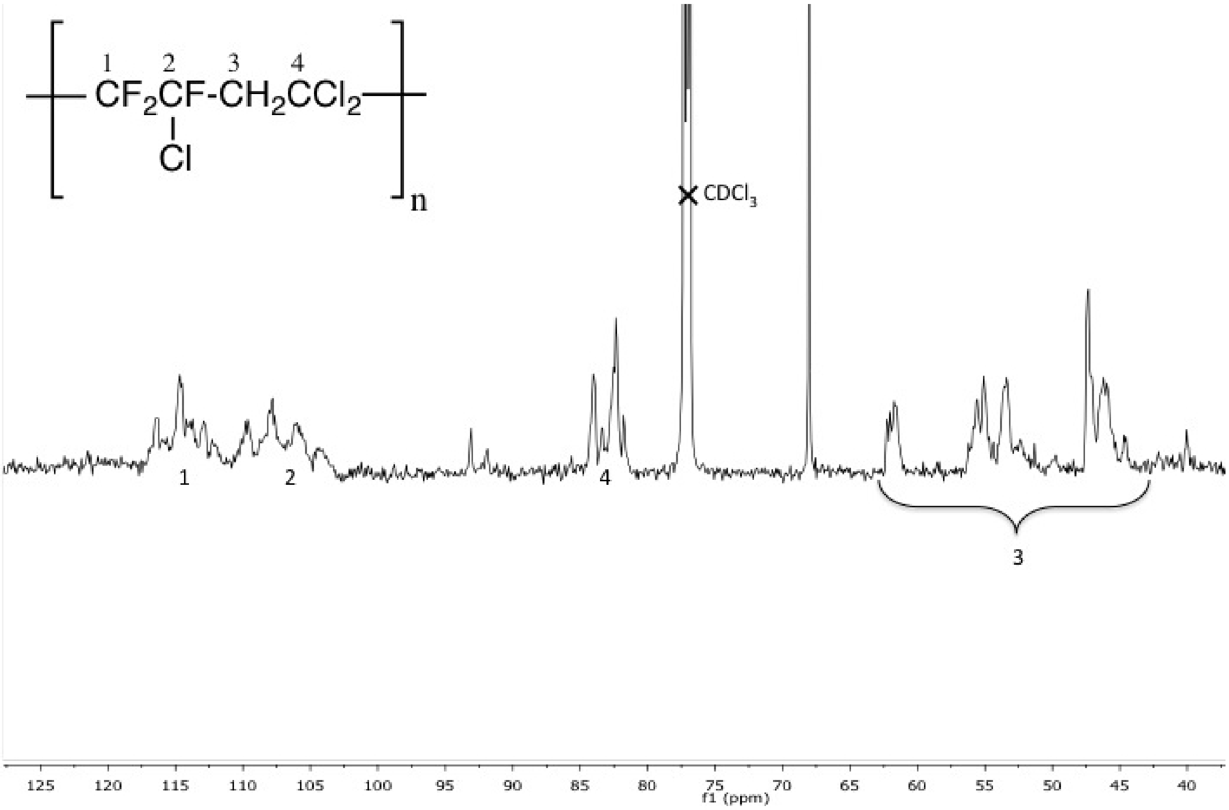

Figure S5. Selected region from the 600 MHz ^{13}C spectrum of poly(CTFE-*co*-VDC) obtained in $CDCl_3$ at room temperature.

Pulse Program Used for ^{1}H{^{13}C} HSQC Experiment

```
/* gHSQC_20141120 - Gradient Selected phase-sensitive HSQC

        Features included:
                F1 Axial Displacement

        Paramters:
                sspul :         selects magnetization randomization option
                nullflg:selects TANGO-Gradient option
                hsglvl:         Homospoil gradient level (DAC units)
                hsgt    :       Homospoil gradient time
                gzlvlE  :       encoding Gradient level
                gtE     :       encoding gradient time
                EDratio         :       Encode/Decode ratio
                gstab   :       recovery delay
                j1xh    :       One-bond XH coupling constant

KrishK-         Last revision   : June 1997
KrishK-         Revised                 : July 2004
KrishK  -       Includes slp saturation option : July 2005
KrishK - includes purge option : Aug. 2006
****v17,v18,v19 are reserved for PURGE ***
Linlin Li, Chun Gao and P Rinaldi - Added decoupler pulse on third channel in middle of
evolution period Nov 2014 to standard gHSQC in version vnmrJ 3.2

*/

#include <standard.h>
#include <chempack.h>


static int      ph1[4] = {1,1,3,3},
                ph2[2] = {0,2,},
                ph3[8] = {0,0,0,0,2,2,2,2},
                ph4[16] = {0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2},
                ph5[16] = {1,3,3,1,3,1,1,3,3,1,1,3,1,3,3,1};

pulsesequence()

{
  double
        pwxlvl   = getval("pwxlvl"),
           pwx2lvl   = getval("pwx2lvl"),
        pwx      = getval("pwx"),
        pwx2      = getval("pwx2"),
        gzlvlE = getval("gzlvlE"),
             gtE = getval("gtE"),
             EDratio = getval("EDratio"),
             gstab = getval("gstab"),
             mult = getval("mult"),
         hsglvl = getval("hsglvl"),
         hsgt = getval("hsgt"),
         tau,d2correction,
         taug;
 int         icosel,
         prgcycle = (int)(getval("prgcycle")+0.5),
             phase1 = (int)(getval("phase")+0.5),
             ZZgsign;

 tau  = 1/(4*(getval("j1xh")));

 if (mult > 0.5)
  taug = 2*tau;
 else
  taug = gtE + gstab + 2*GRADIENT_DELAY;
 ZZgsign=-1;
 if (mult == 2) ZZgsign=1;
 icosel = 1;

assign(ct,v17);
assign(zero,v18);
assign(zero,v19);

if (getflag("prgflg") && (satmode[0] == 'y') && (prgcycle > 1.5))
  {
     hlv(ct,v17);
     mod2(ct,v18); dbl(v18,v18);
     if (prgcycle > 2.5)
        {
           hlv(v17,v17);
           hlv(ct,v19); mod2(v19,v19); dbl(v19,v19);
        }
   }

 settable(t1,4,ph1);
 settable(t2,2,ph2);
 settable(t3,8,ph3);
 settable(t4,16,ph4);
 settable(t5,16,ph5);

getelem(t1, v17, v1);
getelem(t3, v17, v3);
```

```
  getelem(t4, v17, v4);
  getelem(t2, v17, v2);
  getelem(t5, v17, oph);

  assign(zero,v6);
  add(oph,v18,oph);
  add(oph,v19,oph);

/*
   mod2(id2,v14);
   dbl(v14,v14);
*/
   initval(2.0*(double)(((int)(d2*getval("sw1")+0.5)%2)),v14);

   if ((phase1 == 2) || (phase1 == 5))
     icosel = -1;

   add(v2,v14,v2);
   add(oph,v14,oph);

status(A);
   obspower(tpwr);
   dec2power(pwx2lvl);
   delay(5.0e-5);
   if (getflag("sspul"))
        steadystate();

   if (satmode[0] == 'y')
     {
       if ((d1-satdly) > 0.02)
               delay(d1-satdly);
       else
               delay(0.02);
       if (getflag("slpsat"))
          {
               shaped_satpulse("relaxD",satdly,zero);
               if (getflag("prgflg"))
                  shaped_purge(v6,zero,v18,v19);
          }
       else
          {
               satpulse(satdly,zero,rof1,rof1);
               if (getflag("prgflg"))
                  purge(v6,zero,v18,v19);
          }
     }
   else
         delay(d1);

   if (getflag("wet"))
     wet4(zero,one);

   decpower(pwxlvl);

status(B);

    if (getflag("nullflg"))
    {
       rgpulse(0.5*pw,zero,rof1,rof1);
       delay(2*tau);
       simpulse(2.0*pw,2.0*pwx,zero,zero,rof1,rof1);
       delay(2*tau);
       rgpulse(1.5*pw,two,rof1,rof1);
       zgradpulse(hsglvl,hsgt);
       delay(1e-3);
    }

     rgpulse(pw,v6,rof1,rof1);
     delay(tau);
     simpulse(2*pw,2*pwx,zero,zero,rof1,rof1);
     delay(tau);
     rgpulse(pw,v1,rof1,rof1);
         zgradpulse(hsglvl,2*hsgt);
         delay(1e-3);
     decrgpulse(pwx,v2,rof1,2.0e-6);

         d2correction=pw;
         if (2.0*pwx2>pw) d2correction=2.0*pwx2;

     if (d2/2 - d2correction - 6.0e-6 > 0.0)
    delay(d2/2 - d2correction - 6.0e-6);
     else
    delay(d2/2);
     dec2rgpulse(pwx2,zero, 2.0e-6,2.0e-6);
     sim3pulse(2*pw,(double) 0.0,2*pwx2,zero,zero,one,2.0e-6,2.0e-6);
     dec2rgpulse(pwx2,zero, 2.0e-6,2.0e-6);
     if (d2/2 - d2correction - 6.0e-6 > 0.0)
    delay(d2/2 - d2correction - 6.0e-6);
     else
    delay(d2/2);
     zgradpulse(gzlvlE,gtE);
     delay(taug - gtE - 2*GRADIENT_DELAY);
```

```
    simpulse(mult*pw,2*pwx,zero,zero,rof1,rof1);
    delay(taug - (2*pwx/PI) - 2.0e-6);

    decrgpulse(pwx,v4,2.0e-6,rof1);
        zgradpulse(ZZgsign*0.6*hsglvl,1.2*hsgt);
        delay(1e-3);
    rgpulse(pw,v3,rof1,rof1);
    delay(tau - (2*pw/PI) - 2*rof1);
    simpulse(2*pw,2*pwx,zero,zero,rof1, rof2);
    decpower(dpwr);
    dec2power(dpwr2);
    zgradpulse(icosel*2.0*gzlvlE/EDratio,gtE/2.0);
    delay(tau - gtE/2.0 - 2*GRADIENT_DELAY - POWER_DELAY);
  status(C);
}
```