Harnessing open-source technology for low-cost automation in synthesis: flow chemical deprotection of silyl ethers using a homemade autosampling system.

SUPPORTING INFORMATION

M. O'Brien,*[a] L. Konings[ab], M. Martin[a] and J. Heap[a]

a Lennard-Jones Laboratories, School of Chemical and Physical Sciences, Keele University, Keele, Borough of Newcastle-under-Lyme, Staffordshire, United Kingdom, ST5 5BG.

b Avans University of Applied Sciences, Lovensdijkstraat 61-63, 4818 AJ Breda, North Brabant, the Netherlands.

Contents **Page**

General Experimental Information **S2**

General Procedure for the Synthesis of Silyl Ethers **1a-j** **S2**

General Procedure for the Deprotection of Silyl Ethers **1a-j** **S2**

Discussion of Control System **S3**

Image of Overall Flow System Apparatus **S4**

Detail of Mechanical Coupling for Motor Driven 3-Way Valves **S4-S6**

Detail of Autosampler Frame **S6**

NMR Data for Silyl Ethers **1a-j** **S7-S10**

NMR Data for Deprotected Alcohols **3a-j** **S11-S14**

Control Computer Python Script (controller.py) **S15-S25**

Raspberry Pi Computer Python Script (rpi.py) **S26-S31**

^{13}C, DEPT, ^{1}H NMR Spectra of Silyl Ethers **1a-j** **S32-S61**

^{13}C, DEPT, ^{1}H NMR Spectra of Deprotected Alcohols **3a-j** **S62-S91**

General Experimental Information:

Unless specified otherwise, all reagents were purchased from commercial suppliers and were used as received without further purification. Dichloromethane was purchased from Fisher Scientific (D/1850/17 2.5L). Completion of reactions was initially determined using thin layer chromatography (TLC). TLC plates used were Merck Silica-Gel 60 F_{254}, aluminium backed, 1.005554.0001). TLC plates were visualized using vanillin, potassium permanganate, iodine vapor or shortwave ultraviolet light (254 nm).

NMR spectroscopy was carried out on Bruker Sytems (300 MHz: Magnet - Bruker Spectrospin 300 MHz/52 mm, Spectrometer – Avance 300), (400 MHz: Magnet - Bruker Ascend 400, Spectrometer: Avance III 400). $CDCl_3$ was purchased from Cambridge Isotopes, Andover Massachusetts, USA (99.8% D, DLM-7TB-100) and was stored over granular anhydrous potassium carbonate (approx. 5 g added to bottle). NMR data are quoted in parts-per-million (ppm) relative to tetramethylsilane (TMS) at 0 ppm. Proton NMR spectra were calibrated in relation to the residual H solvent peak ($CDCl_3$ = 7.26 ppm) ^{13}C NMR spectra were calibrated in relation to the solvent peak ($CDCl_3$ – central peak of triplet = 77.0 ppm). Multiplicity is indicated by: d-doublet, t-triplet, q-quartet, dd-doublet of doublets etc). Coupling constants (*J*) are given in Hertz (Hz).

Piston-Pump 1 (DCM) was a Knauer Azura HPLC pump, purchased from PMAC Scientific, Aberdeen UK. Piston-Pump 2 (aqueous in) was a Jasco PU-980 HPLC pump. Syringe-pump 1 and 2 were New-Era NE-1000 pumps, purchased from Word Precision Instruments (Alladin AL-1000).

General Procedure for the formation of silyl ethers (1a).

The starting material, 2-(3,4-dimethoxyphenyl)-ethan-1-ol (521 mg, 2.86 mmol, 1.0 equiv) was dissolved in dichloromethane (20 mL) and imidazole (583 mg, 8.58 mmol, 3.0 equiv) was added. The mixture was stirred for 5 minutes. *tert*-Butyldimethylsilyl chloride (518 mg, 3.43 mmol, 1.2 equiv) was added and the reaction mixture was stirred at room temperature for 1 hr. Water (20 mL) and dichloromethane (20 mL) were added and the mixture was shaken in a separating funnel. The aqueous layer was separated and washed with dichloromethane (2 x 5 mL). The combined organic layers were washed with water (20 mL), brine (20 mL) and dried over MgSO4. Following filtration, the solvent was removed under reduced pressure. The crude material was purified by column chromatography on silica gel (eluting with a gradient from petroleum ether to 1:1 petroleum ether : ether) to afford 820 mg of **1a** (97%). ^{1}H and ^{13}C NMR Data provided below.

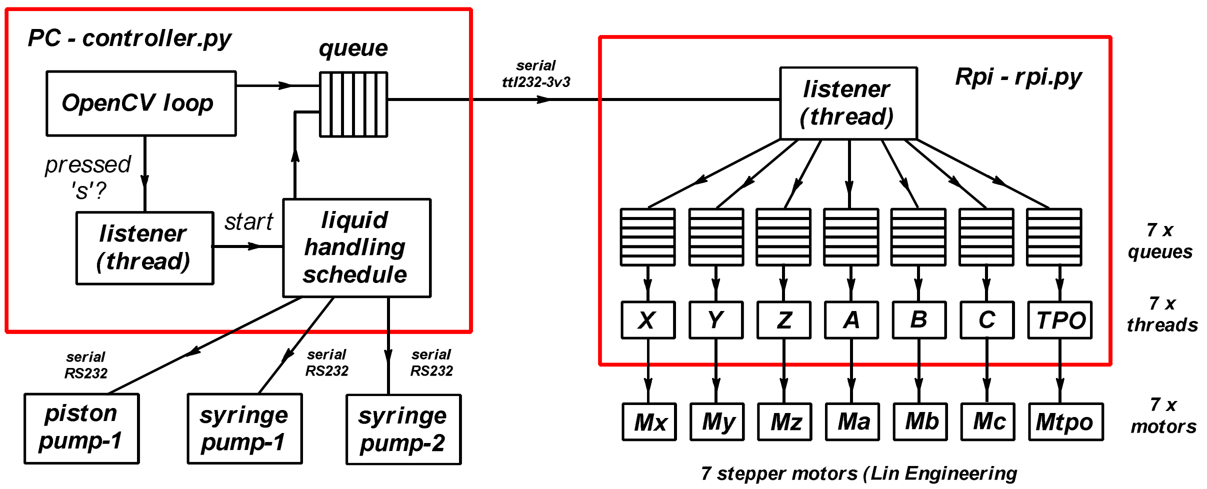
General Procedure for the flow deprotection of silyl ethers (3a-j).

The system (Figure 4) was primed with solvent (DCM and 0.3 M aqueous NaOH) prior to the introduction of the substrates. Substrates were present as 0.034 M solutions (20 mL) in glass vials. These were placed in a square 4 x 4 rack. Following initiation of the computer-vision system (and checking to make sure the aqueous-out tap was opening/closing properly), activation of the autosampler/liquid-handling schedule was initiated by pressing ‘s’ on the computer keyboard. The outlet of the flow stream for each product was then collected until the autosampler moved to the waste position between each run. 5 mL of substrate was taken up during each run, 4 mL of which entered the holding loop (the line between the autosampler and 3-way-valve 1 was 1 mL in volume). Outlet collection flasks were changed manually. The products were isolated by removing solvent under reduced pressure.

Control System

As shown in the schematic below, the control was distributed across a PC (Dell Studio 1558, Windows 7) running the ***controller.py*** script (Python 2.7.8, OpenCV CV2) and a Raspberry Pi (Rpi model 3B v2, Linux: Raspbian Jessie) running the ***rpi.py*** script (python scripts are reproduced below and the .py files are included separately). The PC was connected to the Raspberry Pi using a USB-Serial-UART cable (ttl232-3v3). USB-Serial-RS232 cables were used to connect the PC to piston-pump-1 (Knauer requires null-modem/crossover serial cable) and the two syringe pumps ('normal' serial cable). A Microsoft Lifecam-HD webcam was connected to the PC to monitor the liquid-liquid extraction.

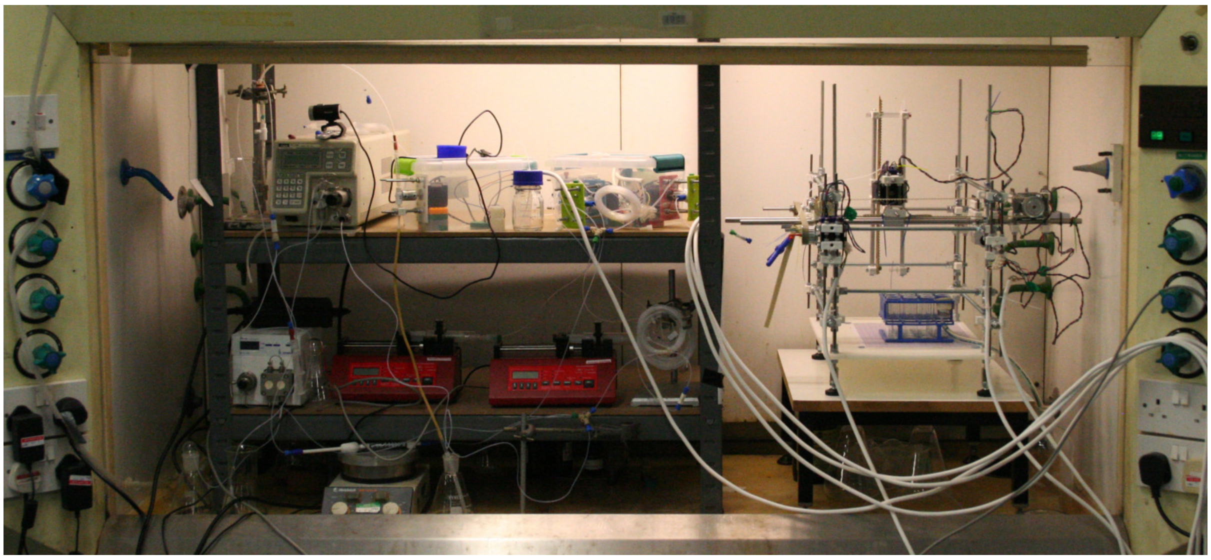
The main thread in rpi.py just reads serial input lines and, depending on the first three letters, places the associated numeric value in one of seven queues. Each queue is read by one of seven reader threads which simply takes the next item in the queue and controls the corresponding motor accordingly, via an Allegro A4988 stepper driver (Pololu breakout board). X, Y and Z correspond to the motors on the 3-axis autosampler. A, B and C correspond to the three 3-way valves and TPO corresponds to the aqueous-out 2-way valve. Following initiation of the system (where the user selects upper and lower bounds for the liquid-liquid interface), the main OpenCV loop dynamically opens and closes the aqueous-out 2-way valve. A 'listener' thread in controller.py is activated from the OpenCV loop when the 's' key is pressed and, in turn, starts the liquid-handling schedule.



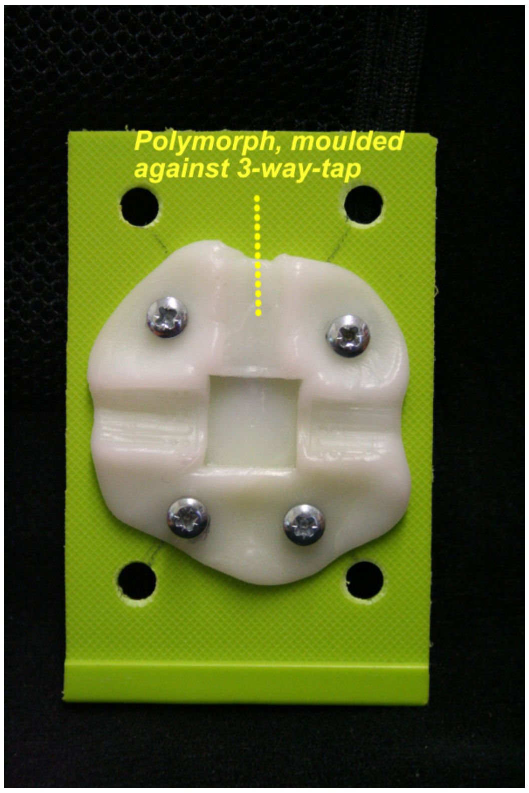
7 stepper motors (Lin Engineering 4118M-01RO) driven via Allegro A4988 motor drivers (Pololu breakout boards). Powered by TENMA PSU (18 V ≈ 2.2 A)

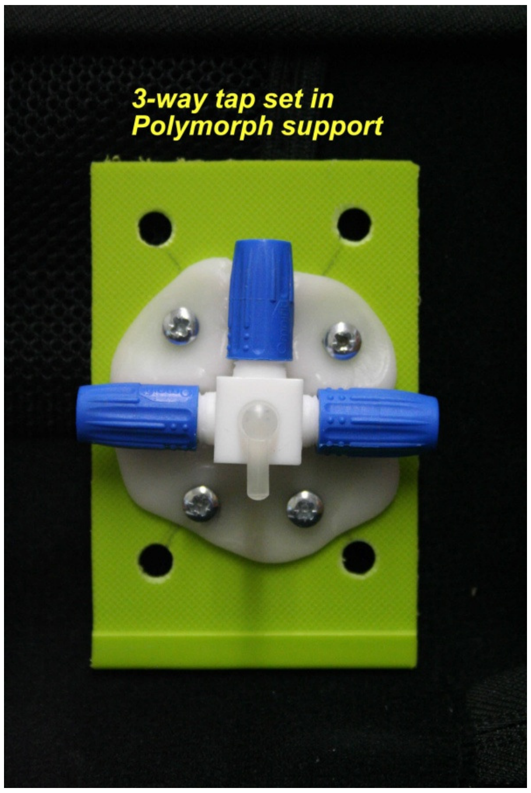
For an explanation of the operation of the autosampler and valve switching, see the separate video file which illustrates the main steps involved for the processing of each substrate sample.

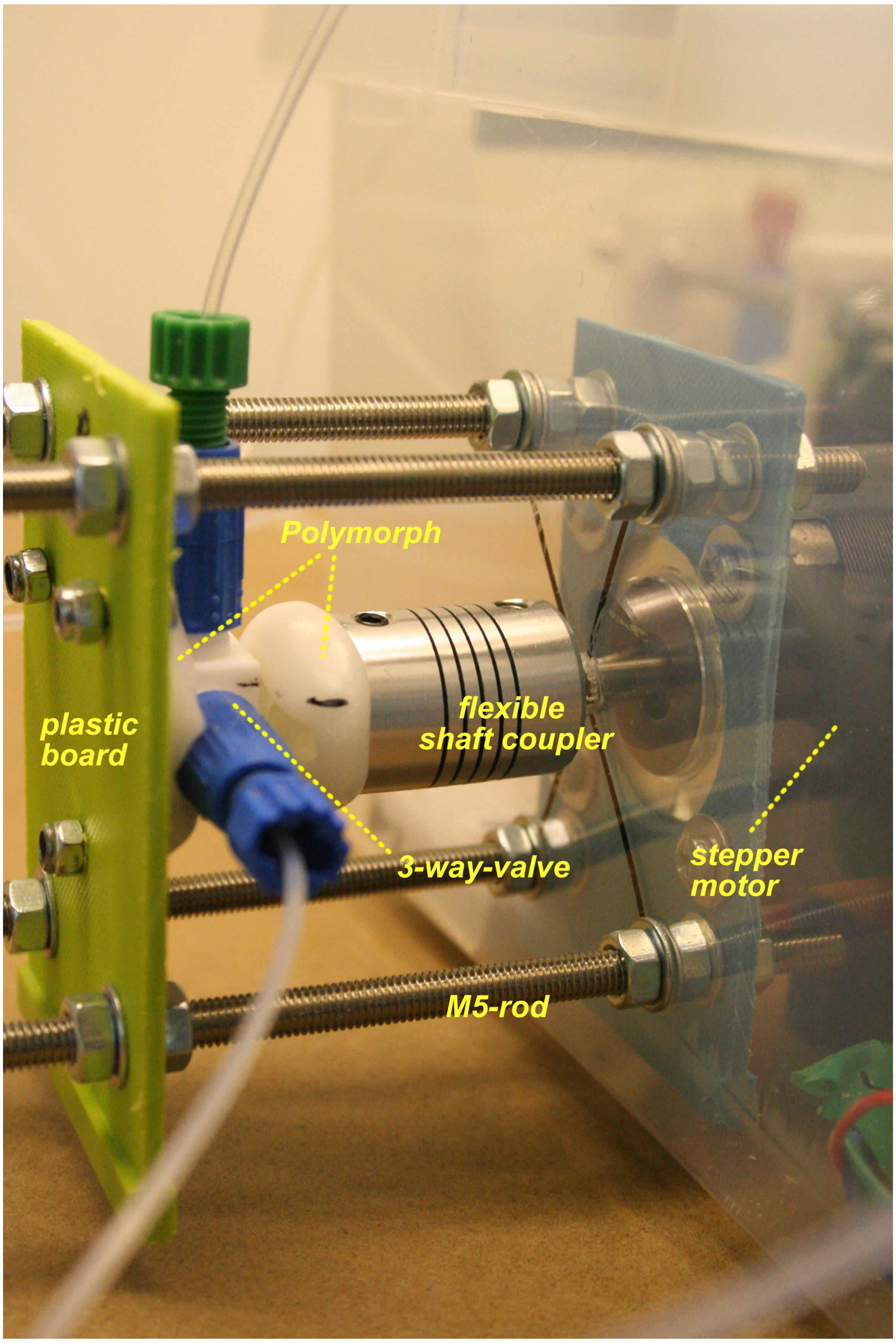
Image of Flow Setup

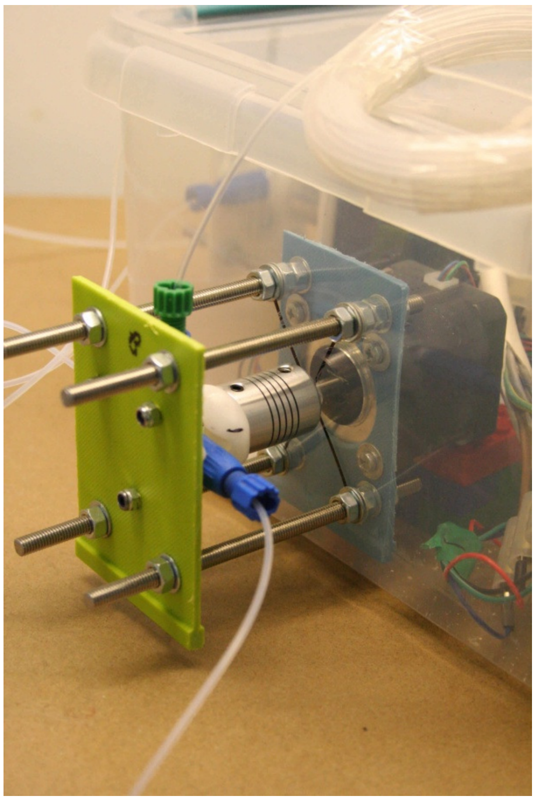
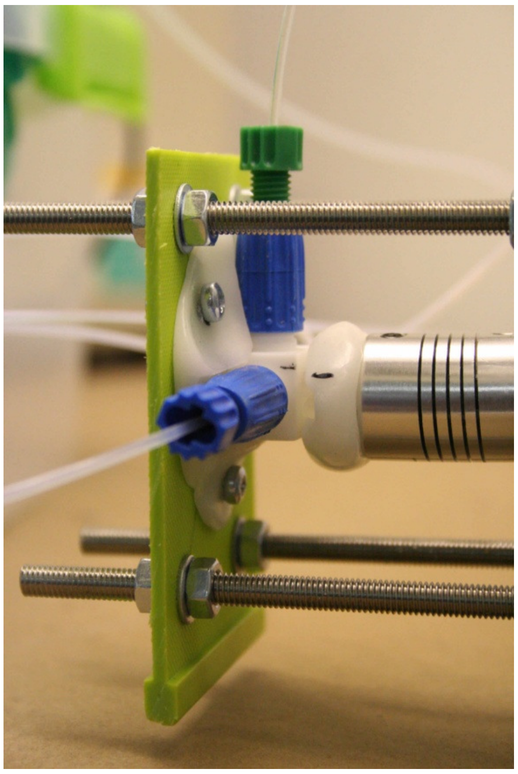


Motor Coupling to 3-way valves:



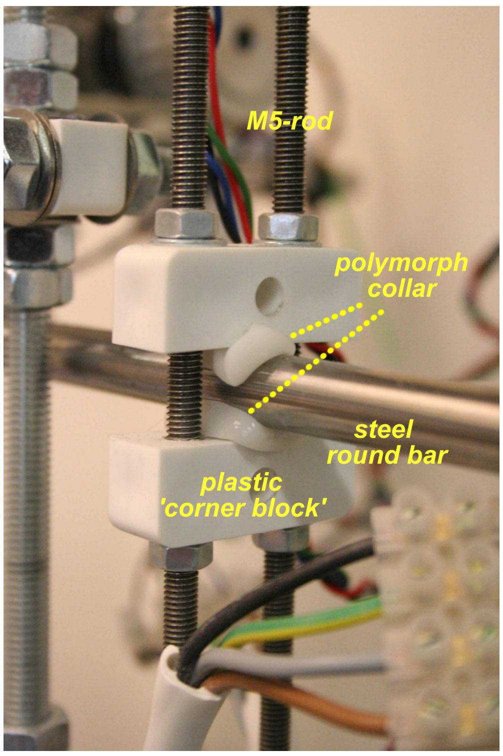


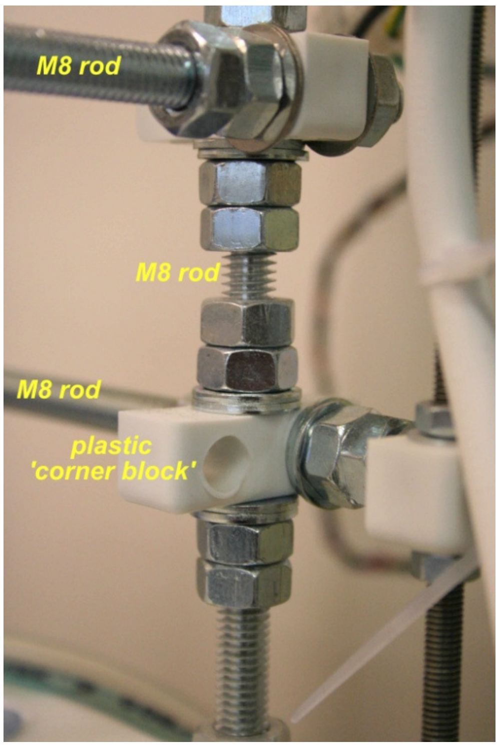

Polymorph
plastic board
flexible shaft coupler
3-way-valve
stepper motor
M5-rod

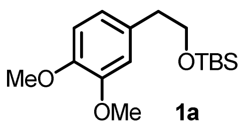
Autosampler frame.

Based on plastic corner blocks and M8/M5 rod. Steel round bar is for bearing trucks to move along. For additional detail of the autosampler see the separate video file showing the movement of the z-axis.





NMR data for compounds 1a-j



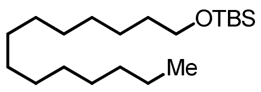
1a

521 mg of the alcohol provided 820 mg of the silyl ether (97%)

^{1}**H NMR** (300 MHz, *$CDCl_3$*) δ ppm 6.83-6.70 (m, 3H), 3.87 (s, 3H), 3.86 (s, 3H), 3.78 (t, *J* = 7.0 Hz, 2H), 2.76 (t, *J* = 7.0 Hz, 2H), 0.88 (s, 9H), -0.01 (s, 6H)

^{13}**C NMR** (75 MHz, *$CDCl_3$*) δ ppm 148.56, 147.32, 131.85, 120.91, 112.42, 110.98, 64.70, 55.87, 55.73, 39.15, 25.91, 18.31, -5.39

F. A. Davis and P. K. Mohanty, *J. Org. Chem.*, 2002, **67**, 1290-1296.



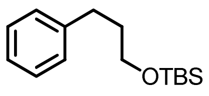
1b

1.10 g of the alcohol provided 1.59 g of the silyl ether (94%)

^{1}**H NMR** (300 MHz, *$CDCl_3$*) δ ppm 3.59 (t, *J* = 6.6 Hz, 2H), 1.55-1.41 (m, 2H), 1.34-1.17 (m, 22H), 0.89 (s, 12H), 0.05 (s, 6H)

^{13}**C NMR** (75 MHz, *$CDCl_3$*) δ ppm 63.35, 32.89, 31.93, 29.66, 29.46, 29.37, 25.98, 25.80, 22.70, 18.38, 14.13, -5.27

J. M. Aizpurua, F. P. Cossio and C. Palomo, *J. Org. Chem.*, 1986, **51**, 4941-4943.



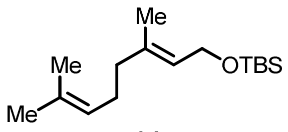
1c

508 mg of the alcohol provided 670 mg of the silyl ether (72%)

^{1}**H NMR** (300 MHz, *$CDCl_3$*) δ ppm 7.38-7.19 (m, 5H), 3.69 (t, *J* = 6.3 Hz, 2H), 2.73 (m, 2H), 1.89 (m, 2H), 0.97 (s, 9H), 0.11 (s, 6H)

^{13}**C NMR** (75 MHz, *$CDCl_3$*) δ ppm 142.23, 128.45, 128.25, 125.64, 62.33, 34.46, 32.08, 25.95, 18.32, -5.29

A. S. Pilcher and P. DeShong, *J. Org. Chem.*, 1993, **58**, 5130-5134.



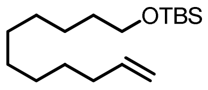
1d

533 mg of the alcohol provided 825 mg of the silyl ether (89%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 5.34-5.26 (m, 1H), 5.14-5.03 (m, 1H), 4.19 (d, *J* = 5.9 Hz, 2H), 2.16-1.94 (m, 4H), 1.68 (s, 3H), 1.62 (s, 3H), 1.60 (s, 3H), 0.90 (s, 9H), 0.07 (s, 6H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 136.86, 131.53, 124.33, 124.07, 60.34, 39.51, 26.31, 26.01, 25.69, 18.43, 17.67, 16.33, -5.06

Y. Imamura, H. Takikawa, M. Sasaki and K. Mori, *Org. Biomol. Chem.*, 2004, **2**, 2236-2244.



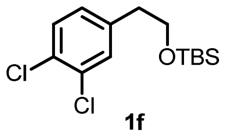
1e

539 mg of the alcohol provided 744 mg of the silyl ether (83%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 5.81 (ddt, *J* = 16.9, 10.2, 6.7 Hz, 1H), 5.03-4.95 (m, 1H), 4.95-4.89 (m, 1H), 3.60 (t, *J* = 6.7 Hz, 2H), 2.04 (dt, *J* = 6.7, 6.7 Hz, 2H), 1.55-1.44 (m, 2H), 1.41-1.23 (m, 12H), 0.89 (s, 9H), 0.05 (s, 6H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 139.22, 114.09, 63.32, 33.82, 32.88, 29.58, 29.43, 29.13, 28.93, 25.98, 25.79, 18.38, -5.27

L. Balas, J. Bertrand-Michel, F. Viars, J. Faugere, C. Lefort, S. Caspar-Bauguil, D. Langin and T. Durand, *Org. Biomol. Chem.*, 2016, **14**, 9012-9020.

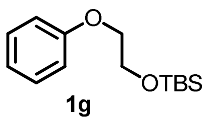


1f

743 mg of the alcohol provided 1.12 g of the silyl ether (94%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.34 (d, *J* = 8.2 Hz, 1H), 7.31 (d, *J* = 2.0 Hz, 1H), 7.04 (dd, *J* = 8.2, 2.0 Hz, 1H), 3.77 (t, *J* = 6.5 Hz, 2H), 2.75 (t, *J* = 6.5 Hz, 2H), 0.86 (s, 9H), -0.03 (s, 6H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 139.72, 131.90, 131.16, 129.99, 129.96, 128.64, 63.65, 38.47, 25.83, 18.24, -5.50



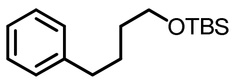
1g

1.12 g of the alcohol provided 1.99 g of the silyl ether (97%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.32-7.24 (m, 2H), 6.98-6.88 (m, 3H), 4.06-3.95 (m, 4H), 0.91 (s, 9H), 0.10 (s, 6H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 158.87, 129.39, 120.65, 114.49, 69.09, 62.01, 25.91, 18.41, -5.21

B. A. D'Sa, D. McLeod and J. G. Verkade, *J. Org. Chem.*, 1997, **62**, 5057-5061.



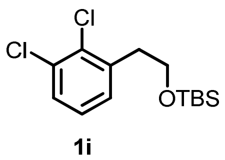
1h

1.00 g of the alcohol provided 1.59 g of the silyl ether (90%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.32-7.25 (m, 2H), 7.22-7.15 (m, 3H), 3.63 (t, *J* = 6.3 Hz, 2H), 2.63 (t, *J* = 7.5 Hz, 2H), 1.76-1.50 (m, 4H), 0.90 (s, 9H), 0.05 (s, 6H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 142.62, 128.39, 128.23, 125.61, 63.00, 35.67, 32.41, 27.65, 25.96, 18.36, -5.29

C. Chen, T. R. Dugan, W. W. Brennessel, D. J. Weix and P. L. Holland, *J. Amer. Chem. Soc.*, 2014, **136**, 945-955.

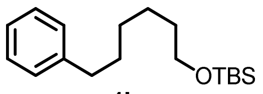


1i

609 mg of the alcohol provided 906 mg of the silyl ether (93%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.33 (dd, *J* = 7.7, 1.9 Hz, 1H), 7.17 (dd, *J* = 7.7, 1.9 Hz, 1H), 7.11 (dd, *J* = 7.7, 7.7 Hz, 1H), 3.83 (t, *J* = 6.8 Hz, 2H), 3.00 (t, *J* = 6.8 Hz, 2H), 0.85 (s, 9H), -0.03 (s, 6H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 139.07, 132.93, 132.29, 129.81, 128.47, 126.81, 62.03, 38.01, 25.86, 18.26, -5.50



1j

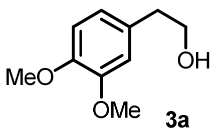
215 mg of the alcohol provided 320 mg of the silyl ether (90%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.33-7.14 (m, 5H), 3.64-3.51 (m, 2H), 2.66-2.55 (m, 2H), 1.66-1.43 (m, 4H), 1.41-1.30 (m, 4H), 0.89 (s, 9H), 0.04 (s, 6H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 142.83, 128.38, 128.20, 125.55, 63.25, 35.90, 32.77, 31.51, 29.08, 25.97, 25.66, 18.37, -5.27

F. González-Bobes and G. C. Fu, *J. Amer. Chem. Soc.*, 2006, **128**, 5360-5361.

NMR data for compounds 3a-j



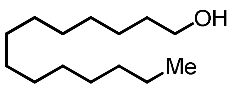
3a

24 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (97%)

^{1}**H NMR** (300 MHz, *CDCl$_3$*) δ ppm 6.84-6.73 (m, 3H), 3.87 (s, 3H), 3.85 (s, 3H), 3.90-3.79 (m, 2H), 2.80 (t, *J* = 6.46, 6.46 Hz, 2H), 1.63 (s, 1H)

^{13}**C NMR** (75 MHz, *CDCl$_3$*) δ ppm 148.86, 147.55, 130.87, 120.83, 112.03, 111.20, 63.67, 55.83, 55.74, 38.65

S. T. Handy, Y. Zhang and H. Bregman, *J. Org .Chem.*, 2004, **69**, 2362-2366.



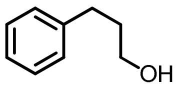
3b

28 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (96%)

^{1}**H NMR** (300 MHz, *CDCl$_3$*) δ ppm 3.63 (t, *J* = 6.6 Hz, 2H), 1.61-1.50 (m, 2H), 1.41-1.18 (m, 23H), 0.91-0.84 (m, 3H)

^{13}**C NMR** (75 MHz, *CDCl$_3$*) δ ppm 63.06, 32.77, 31.91, 29.65, 29.60, 29.42, 29.35, 25.71, 22.68, 14.12

D. Peng, M. Zhang and Z. Huang, *Chem. – Eur. J.,* 2015, **21**, 14737-14741.



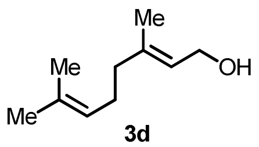
3c

17 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (92%)

^{1}**H NMR** (300 MHz, *CDCl$_3$*) δ ppm 7.34-7.16 (m, 5H), 3.68 (t, *J* = 6.4 Hz, 2H), 2.75-2.68 (m, 2H), 1.95-1.85 (m, 2H), 1.41 (s, 1H)

^{13}**C NMR** (75 MHz, *CDCl$_3$*) δ ppm 141.77, 128.40, 128.37, 125.84, 62.25, 34.19, 32.03

D. Peng, M. Zhang and Z. Huang, *Chem. – Eur. J.,* 2015, **21**, 14737-14741.



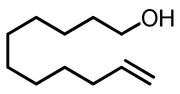
3d

19 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (91%)

^{1}**H NMR** (300 MHz, *$CDCl_3$*) δ ppm 5.42 (tq, *J* = 6.9, 1.9 Hz, 1H), 5.13-5.05 (m, 1H), 4.16 (d, *J* = 6.9 Hz, 2H), 2.16-1.96 (m, 4H), 1.68 (s, 6H), 1.60 (s, 3H)

^{13}**C NMR** (75 MHz, *$CDCl_3$*) δ ppm 139.83, 131.76, 123.84, 123.23, 59.41, 39.52, 26.34, 25.69, 17.69, 16.26

J. Du, G. Xu, H. Lin, G. Wang, M. Tao and W. Zhang, *Green Chem.*, 2016, **18**, 2726-2735.

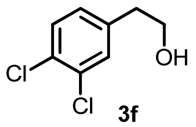


3e

22 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (95%)

^{1}**H NMR** (300 MHz, *$CDCl_3$*) δ ppm 5.75 (ddt, *J* = 17.0, 10.2, 6.7 Hz, 1H), 4.92 (d, *J* = 17.0 Hz, 1H), 4.86 (d, *J* = 10.2 Hz, 1H), 3.57 (t, *J* = 6.1 Hz, 2H), 1.97 (dt, *J* = 6.7, 6.7 Hz 2H), 1.57-1.42 (m, 2H), 1.35-1.16 (m, 14H)

^{13}**C NMR** (75 MHz, *$CDCl_3$*) δ ppm 139.21, 114.09, 63.04, 33.79, 32.75, 29.52, 29.39, 29.09, 28.88, 25.69
M. Szostak, M. Spain and D. J. Procter, *Org. Lett.*, 2012, **14**, 840-843.



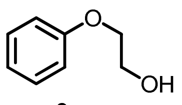
3f

24 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (92%)

^{1}**H NMR** (300 MHz, *$CDCl_3$*) δ ppm 7.37 (d, *J* = 8.2 Hz, 1H), 7.33 (d, *J* = 2.0 Hz, 1H), 7.07 (dd, *J* = 8.2, 2.0 Hz, 1H), 3.87-3.80 (m, 2H), 2.81 (t, *J* = 6.4 Hz, 2H), 1.58 (s, 1H)

^{13}**C NMR** (75 MHz, *$CDCl_3$*) δ ppm 138.92, 132.33, 130.89, 130.40, 130.35, 128.45, 63.06, 38.10

C. A. Mosley, S. J. Myers, E. E. Murray, R. Santangelo, Y. A. Tahirovic, N. Kurtkaya, P. Mullasseril, H. Yuan, P. Lyuboslavsky, P. Le, L. J. Wilson, M. Yepes, R. Dingledine, S. F. Traynelis and D. C. Liotta, *Bioorg. Med. Chem.*, 2009, **17**, 6463-6480.



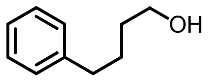
3g

18 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (96%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.34-7.27 (m, 2H), 7.01-6.90 (m, 3H), 4.11-4.07 (m, 2H), 4.01-3.93 (m, 2H), 2.09 (t, *J* = 6.0 Hz, 1H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 158.52, 129.52, 121.10, 114.47, 68.96, 61.49

G. Urgoitia, R. SanMartin, M. T. Herrero and E. Domínguez, *Adv. Synth. & Catal.*, 2016, **358**, 3307-3312



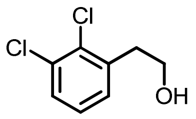
3h

20 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (98%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.33-7.24 (m, 2H), 7.22-7.14 (m, 3H), 3.65 (t, *J* = 6.3 Hz, 2H), 2.64 (t, *J* = 7.4 Hz, 2H), 1.76-1.53 (m, 4H), 1.39 (s, 1H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 142.28, 128.37, 128.26, 125.71, 62.78, 35.59, 32.27, 27.52

T. V. Q. Nguyen, W.-J. Yoo and S. Kobayashi, *Adv. Synth. & Catal.*, 2016, **358**, 452-458.

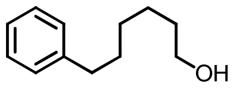


3i

25 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (96%)

^{1}H NMR (300 MHz, *$CDCl_3$*) δ ppm 7.35 (dd, *J* = 7.6, 2.0 Hz, 1H), 7.20 (dd, *J* = 7.6, 2.0 Hz, 1H), 7.15 (dd, *J* = 7.6 Hz, 7.6 Hz, 1H), 3.89 (dt, *J* = 6.5, 6.5 Hz, 2H), 3.05 (t, *J* = 6.5 Hz, 2H), 1.54 (m, 1H)

^{13}C NMR (75 MHz, *$CDCl_3$*) δ ppm 138.48, 133.27, 132.40, 129.37, 128.78, 127.10, 61.62, 37.72



3j

24 mg of the alcohol was formed from 4 mL of a 0.034 M solution of the silyl ether (99%)

^{1}**H NMR** (300 MHz, *$CDCl_3$*) δ ppm 7.32-7.23 (m, 2H), 7.22-7.13 (m, 3H), 3.62 (t, *J* = 6.0 Hz, 2H), 2.62 (d, *J* = 7.5 Hz, 2H), 1.69-1.49 (m, 4H), 1.43-1.33 (m, 5H)

^{13}**C NMR** (75 MHz, *$CDCl_3$*) δ ppm 142.69, 128.35, 128.20, 125.57, 62.93, 35.84, 32.64, 31.42, 29.02, 25.56

R. Suzuki, S. Fuse and H. Tanaka, *Eur. J. Org. Chem.*, 2016, **2016**, 3478-3481.

```
#CONTROLLER.py this script runs on the control system

import cv2
import numpy as np
import time
import serial
import threading
from Queue import Queue
import sys

dim = 30 #this is the relative horizontal dimension of the rack size
step = 0.0318
rack = [(1,1),(1,2), (1,3), (1,4),
        (2,4), (2,3), (2,2), (2,1),
        (3,1), (3,2), (3,3), (3,4),
        (4,4), (4,3), (4,2), (4,1), (1,1)]

#these are the cartesian positions, in sequence, of the vials in the rack
#this list is for a symmetrical 4 x 4 rack, iterating through 15 samples
#samples are taken in a 'snaking' order
#Waste position is 1,1
#needle should be above 1st substrate position (1,2) at start of run



camport = 0 # the camera port, change to select correct camera

cam = cv2.VideoCapture(camport)

ramp_frames = 20   # effectively discards the first 20 frames

posflag = 'high'
tapflag = 'closed'

startflag = False

drag_start = None
drag_end = None
box = (0,0,0,0)

interface = 0.5 #interface position as a proportion between upper and
lower limits

kernel = np.ones((5,5),np.uint8)
kernel2 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))

huevar = 20  # the hue variance for image filtering, adjust accordingly

rpos = None
```

```
ser = serial.Serial() #to rpi
ser.baudrate = 9600
ser.port = 20
ser.open()

ser_sp1 = serial.Serial()#to syringe pump 1
ser_sp1.baudrate = 9600
ser_sp1.port = 23
ser_sp1.open()
ser_sp1.write('00 rat 2\x0D')
time.sleep(0.1)

ser_sp2 = serial.Serial()#to syringe pump 2
ser_sp2.baudrate = 9600
ser_sp2.port = 24
ser_sp2.open()
ser_sp2.write('01 rat 2\x0D')
time.sleep(0.1)

ser_p1 = serial.Serial() # to piston pump 1 (Knauer Azura)
ser_p1.baudrate = 9600
ser_p1.port = 43
ser_p1.open()
ser_p1.write('flow 500 \x0D')
time.sleep(0.1)

myq = Queue(maxsize=0) #this queue is for serial communication to rpi

def dostuff():   # enters this function, in a thread, once 's' pressed
    for i in range(len(rack)-1):
        if i != 0:
            print i
            print 'at position: ' + str(rack[i])

            myq.put('mvz-2000\n')
            print 'lowering needle'
            time.sleep(5)

            #activate syringes here (taps already set).

            #sp1-take in x mL

            ser_sp1.write('00 rat 2 \x0D')
            time.sleep(0.1)
            ser_sp1.write('00 rat 2 \x0D')
            time.sleep(0.1)
            #sending commands to pumps twice avoided occasional drop-outs
            ser_sp1.write('00 dir wdr \x0D')
            time.sleep(0.1)
            ser_sp1.write('00 dir wdr \x0D')
            time.sleep(0.1)
```

```
print 'changing sp1 to wdr'

ser_sp1.write('00 run \x0D')
time.sleep(0.1)
ser_sp1.write('00 run \x0D')
time.sleep(0.1)
print 'taking into sp1'

time.sleep(150) #take in 5ml (2 x 2.5 min)

#sp1-stop:
ser_sp1.write('00 stp \x0D')
time.sleep(0.1)
ser_sp1.write('00 stp \x0D')
time.sleep(0.1)
print 'stopped sp1'

time.sleep(10) # allows pressure to balance in lines

#tp2 to p-1
myq.put('twb 1600\n')
print 'tp2 to p-1' #connects holding loop to piston-pump 1
time.sleep(5)
#tp1 to reactor
myq.put('twa 1600\n') #connects holding loop to reaction loop
(T-1)
print 'tp1 to reactor'
time.sleep(5)
#sp2-take in 7 mL:
ser_sp2.write('01 rat 2 \x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 2 \x0D')
time.sleep(0.1)
ser_sp2.write('01 dir wdr \x0D')
time.sleep(0.1)
ser_sp2.write('01 dir wdr \x0D')
time.sleep(0.1)
print 'changing sp2 to wdr'
time.sleep(2)
ser_sp2.write('01 run \x0D')
time.sleep(0.1)
ser_sp2.write('01 run \x0D')
time.sleep(0.1)
print 'taking in to sp2'
time.sleep(210) #take in 7ml (2 x 3.5 min)
#sp2-stop:
ser_sp2.write('01 stp \x0D')
time.sleep(0.1)
ser_sp2.write('01 stp \x0D')
time.sleep(0.1)
```

```
print 'stopped sp2'
time.sleep(2)

#tp3 to reactor:
myq.put('twc 1600\n')
print 'tp3 to reactor' #connect syringe pump 2 to reactor (T-1)
time.sleep(5)

#start pumping from syringe pump 2:


ser_sp2.write('01 rat 0.5\x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 0.5\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir inf \x0D')
time.sleep(0.1)
ser_sp2.write('01 dir inf \x0D')
time.sleep(0.1)
print 'changing sp2 to infuse'

ser_sp2.write('01 run \x0D')
time.sleep(0.1)
ser_sp2.write('01 run \x0D')
time.sleep(0.1)
print 'pumping from sp2'
time.sleep(2) #gives TsOH solution a slight lead over substrate

#start piston-pump 1:
ser_p1.write('flow 500 \x0D')
time.sleep(0.1)
ser_p1.write('flow 500 \x0D')
time.sleep(0.1)
ser_p1.write('on \x0D')
time.sleep(0.1)
ser_p1.write('on \x0D')
time.sleep(0.1)
print 'starting p1'
time.sleep(835) #7ml 0.5 x 14 min
#stop syringe pump 2, piston pump 1 continues:
ser_sp2.write('01 stp \x0D')
time.sleep(0.1)
ser_sp2.write('01 stp \x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 2\x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 2\x0D')
time.sleep(0.1)
print 'stopped sp2'
#tp3 to reservoir (now ready for next round):
myq.put('twc -1600\n')
```

```
print 'tp3 to reservoir'
print 'reactants in system'

#increase speed of piston pump 1 to 1mL/min:
ser_p1.write('flow 1000 \x0D')
time.sleep(0.1)
ser_p1.write('flow 1000 \x0D')
time.sleep(0.1)
print 'increasing speed of p1'
time.sleep(3600) #continue to pump for 1 hr
#stop piston pump 1:
ser_p1.write('off \x0D')
time.sleep(0.1)
ser_p1.write('off \x0D')
time.sleep(0.1)
#piston pump 1 back to 0.5 mL/min:
ser_p1.write('flow 500 \x0D')
time.sleep(0.1)
ser_p1.write('flow 500 \x0D')
time.sleep(0.1)
print 'stopping p1'
print 'lowering speed of p1'
time.sleep(2)
# move autosampler to waste position:
myq.put('mvz2000\n')
print 'raising needle'
time.sleep(5)

deltax = rack[i][0]- 1
print 'deltax = ' + str(deltax)
deltay = rack[i][1]- 1
print 'deltay = ' + str(deltay)
xsteps = int(dim*deltax/step)
print 'xsteps = ' + str(xsteps)
ysteps = int(dim*deltay/step)
print 'ysteps = ' + str(ysteps)
time.sleep(5)
print 'moving to waste'
if abs(xsteps)>0.1:
        t = ('mvx%s' % str(xsteps))
        t = t + '\n'
        myq.put(t)
        print 'moving x'
        time.sleep((abs(xsteps))/400.0)

if abs(ysteps)>0.1:
        t = ('mvy%s' % str(ysteps))
        t = t + '\n'
        print t
        myq.put(t)
        print 'moving y'
```

```
                time.sleep((abs(ysteps))/400.0)

        #move 3-way-valve 1 to autosampler:
        myq.put('twa -1600\n')
        print 'changing tp1 to autosampler'
        time.sleep(5)

        #switch tp2 to sp-1 for flush (and to reset sp1)
        myq.put('twb -1600\n')
        print 'tp2 to sp1'
        time.sleep(5)
        #start sp1
        ser_sp1.write('00 rat 2 \x0D')
        time.sleep(0.1)
        ser_sp1.write('00 rat 2 \x0D')
        time.sleep(0.1)
        ser_sp1.write('00 dir inf \x0D')
        time.sleep(0.1)
        ser_sp1.write('00 dir inf \x0D')
        time.sleep(0.1)
        print 'changing sp1 to inf'
        time.sleep(2)
        #start pumping from p-1 (flush line to waste)
        ser_sp1.write('00 run \x0D')
        time.sleep(0.1)
        ser_sp1.write('00 run \x0D')
        time.sleep(0.1)
        print 'flusing selector with sp1'
        time.sleep(150) #5ml 2 x 2.5 min
        ser_sp1.write('00 stp \x0D')
        time.sleep(0.1)
        ser_sp1.write('00 stp \x0D')
        time.sleep(0.1)
        print 'stopping sp1'
        time.sleep(5)


    #go to next selector position:
    deltax = 1 - rack[i+1][0]
    print 'deltax = ' + str(deltax)
    deltay = 1 - rack[i+1][1]
    print 'deltay = ' + str(deltay)
    xsteps = int(dim*deltax/step)
    print 'xsteps = ' + str(xsteps)
    ysteps = int(dim*deltay/step)
    print 'ysteps = ' + str(ysteps)
    print 'moving to next selector position'

    if abs(xsteps)>0.1:
            t = ('mvx%s' % str(xsteps))
            t = t + '\n'
```

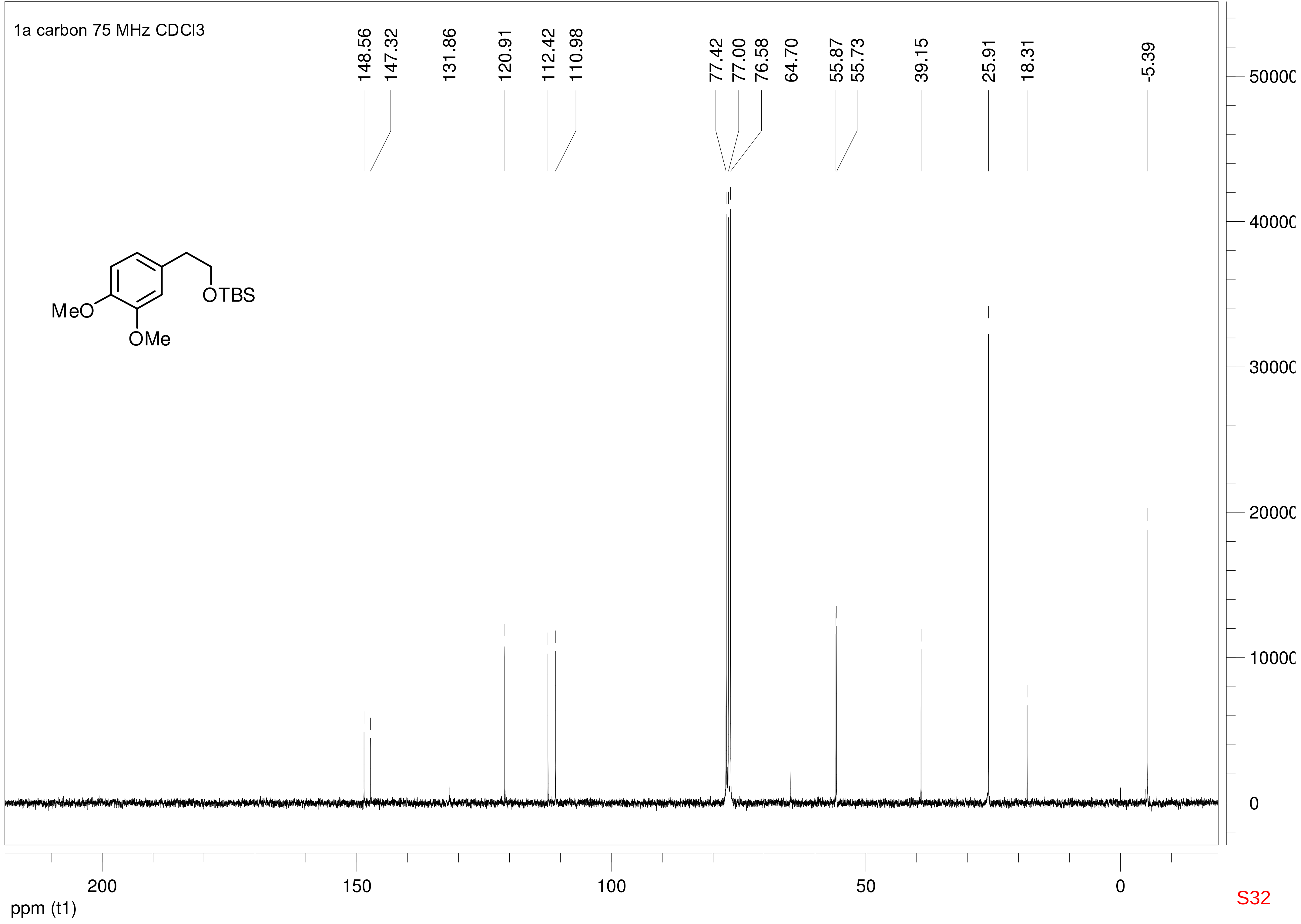
```
                myq.put(t)
                print 'moving x'
                time.sleep((abs(xsteps))/400.0)
        if abs(ysteps)>0.1:
                t = ('mvy%s' % str(ysteps))
                t = t + '\n'
                print t
                myq.put(t)
                print 'moving y'
                time.sleep((abs(ysteps))/400.0)
    print 'returning'
    return


listenflag = True

def listenfunc(): # runs in thread to see if 's' has been pressed
    global listenflag
    while listenflag:
        if startflag:
            print 'startflag'
            dostuff()   # starts autosampler system
            listenflag = False

serflag = True

def myserial():   #writes serial to rpi from myq queue
    #print 'myserial'
    global serflag
    while serflag:
        ser.write(myq.get())
        myq.task_done()
    #print 'returning myserial'
    return


mythread2 = threading.Thread(target = myserial)

listenthread = threading.Thread(target = listenfunc)

def poscalc(refpt1, refpt2, scanpt):

    percent = (float(scanpt[1])-float(refpt2[1]))/(float(refpt1[1])-float(
    refpt2[1]))
    return percent

def posdo(fract):


    global ser
    global tapflag
    global posflag

    if posflag == 'high':
        if fract < 0:
            if tapflag == 'closed':
                pass
            if tapflag == 'open':
                tapflag = 'closed'
                this = (str('tpo-888').zfill(1) + '\n')
                myq.put(this)
                print 'close tap'
        if fract > 1:
            posflag = 'low'
    if posflag == 'low':
        if fract > 1:
            if tapflag == 'open':
                pass
            if tapflag == 'closed':
                tapflag = 'open'
                this = (str('tpo888').zfill(1) + '\n')
                myq.put(this)
                print 'open tap'
        if fract < 0:
            posflag = 'high'


def trackcam(cam, hueval): #this is the main OpenCV function
    global huevar
    global startflag
    global serflag

    hue1 = hueval - huevar
    hue2 = hueval + huevar

    go = True
    while go:
        retval, f = cam.read()
        hsv = cv2.cvtColor(f, cv2.COLOR_BGR2HSV)
        lower = np.array([hue1, 50, 50])
        higher = np.array([hue2, 255, 255])
        mask = cv2.inRange(hsv, lower, higher)
        opened2 = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel2)
        _, contours, heirarchy = cv2.findContours(opened2, cv2.RETR_TREE,
        cv2.CHAIN_APPROX_SIMPLE)
        areas = [cv2.contourArea(c) for c in contours]

        if len(areas) > 0:

            max_index = np.argmax(areas)
            cnt = contours[max_index]
            M = cv2.moments(cnt)
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
            cv2.circle(f, (cx, cy), 10, (255,255,0), 3)

        cv2.imshow('mask', mask)

        fract = poscalc(refpos2, refpos1, (cx, cy))
        posdo(fract)

        cv2.circle(f, refpos1, 5, (255,100,100), thickness = 2)
        cv2.circle(f, refpos2, 5, (255,100,100), thickness = 2)

        cv2.imshow('frame', f)
        l = cv2.waitKey(5)
        if l == ord('s'):
            print 'pressed s'
            startflag = True #initiates starting the autosampler 'do
            stuff' thread'
        if l == 27:
            cv2.destroyAllWindows()
            serflag = False
            break


def takepic():
    retval, im = cam.read()
    return im

def on_mouse2(event, x, y, flags, param):
    global rpos
    if event == cv2.EVENT_LBUTTONDOWN:
        rpos = (x,y)

def getrefpos(image):
    global rpos
    rpos = None
    im = image
    cv2.namedWindow('select refpos', 1)
    cv2.setMouseCallback('select refpos', on_mouse2)
    a = True
    while a:
        if rpos:
            a = False
            cv2.destroyWindow('select refpos')
            return rpos
        cv2.imshow('select refpos', im)
        k = cv2.waitKey(7)
        if k == 27:
            cv2.destroyWindow('select refpos')
            return None
```

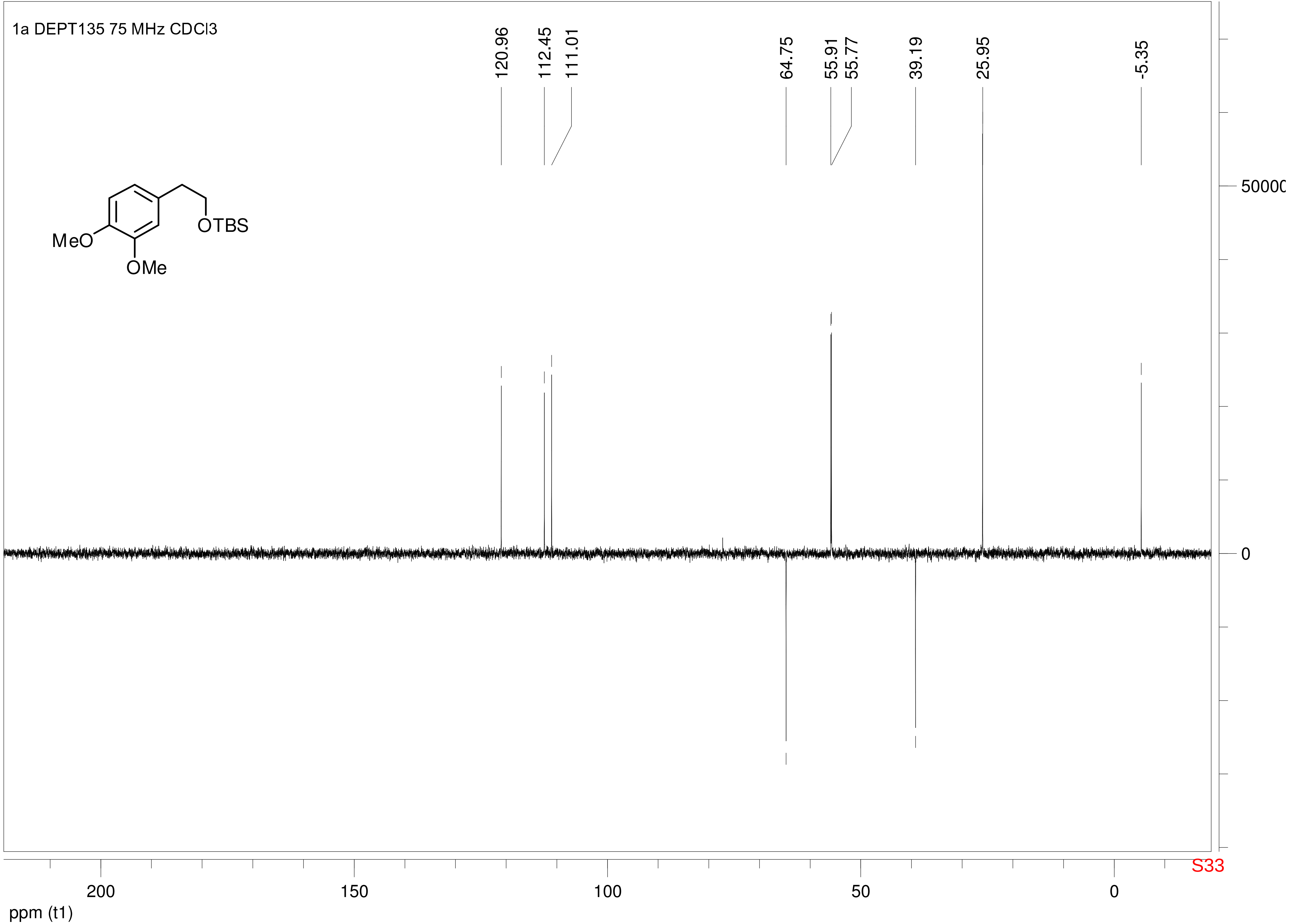
```
def on_mouse(event, x, y, flags, param):
    global drag_start
    global drag_end
    global box
    if event == cv2.EVENT_LBUTTONDOWN:
        drag_end = False
        drag_start = (x,y)
        print drag_start
        print 'left down'
    if event == cv2.EVENT_LBUTTONUP:
        drag_end = (x,y)
        print drag_end
        print 'left up'

    if drag_start and drag_end:

        xmin = min (drag_start[0], drag_end[0])
        ymin = min (drag_start[1], drag_end[1])
        xmax = max (drag_start[0], drag_end[0])
        ymax = max (drag_start[1], drag_end[1])
        box = (xmin, ymin, xmax-xmin, ymax-ymin)
    if box:
        print box

def getbox(image):
    print getbox
    im = image
    cv2.namedWindow('select', 1)
    cv2.setMouseCallback('select', on_mouse)
    a = True
    while a:
        if box[2]>0 and box[3]>0:
            cv2.destroyWindow('select')
            return box
        cv2.imshow('select', im)
        k = cv2.waitKey(7)
        if k == 27:
            cv2.destroyWindow('select')
            return None

def gethist(image):
    global box
    im = image
    im2 = im[box[1]:(box[1] + box[3]), box[0]: (box[0]+box[2])]
    hsv = cv2.cvtColor(im2, cv2.COLOR_BGR2HSV)
    hist = cv2.calcHist([hsv], [0], None, [180], [0, 180])

    return hist
```

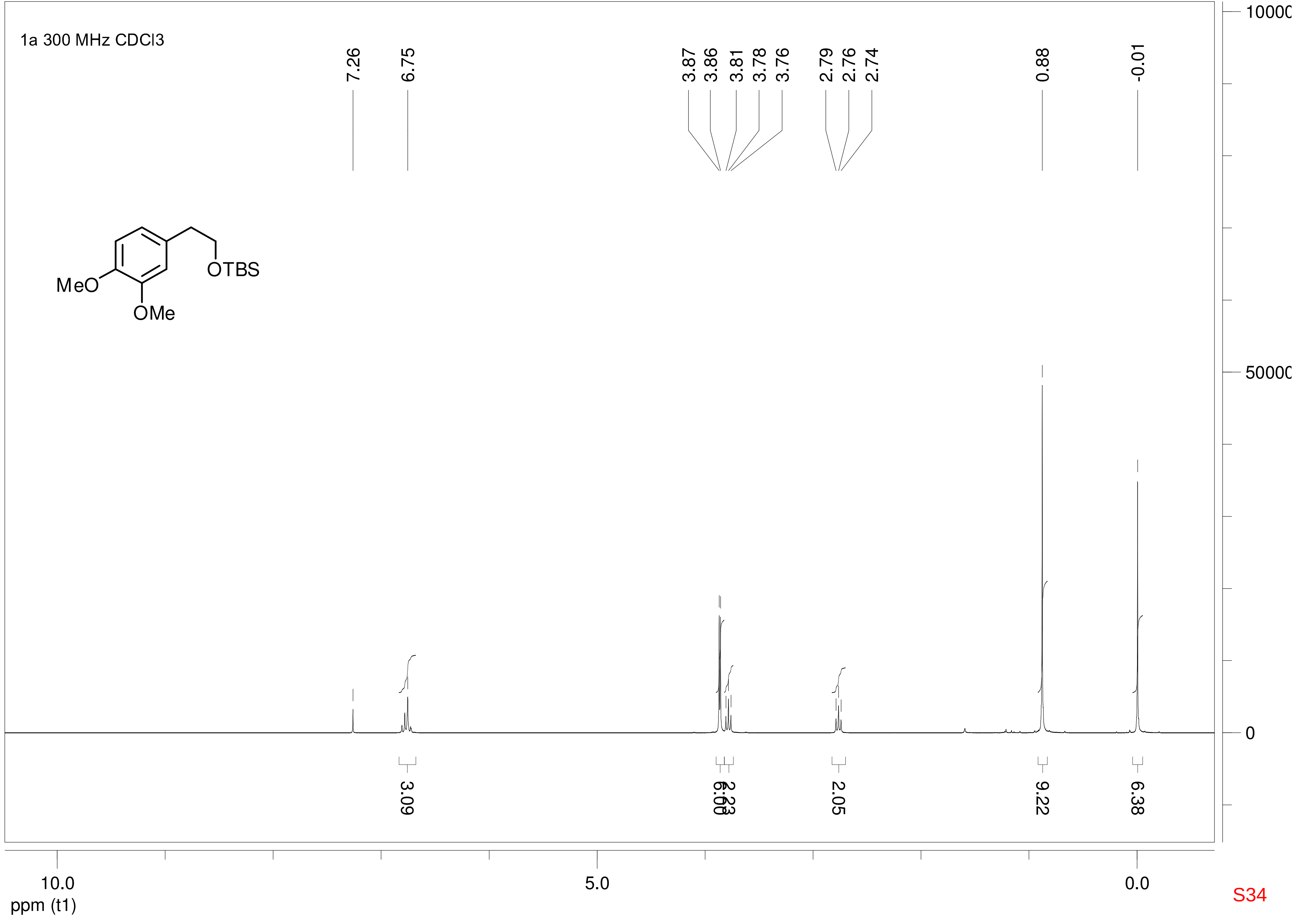
```
for i in range(ramp_frames):
    temp = takepic()

capture = takepic()

refpos1 = getrefpos(capture)
print refpos1
refpos2 = getrefpos(capture)
print refpos2

box1 = getbox(capture)
print box1


hist = gethist(capture)

m = np.argmax(hist)
print 'strongest hue is ' + str(m)

mythread2.start()
listenthread.start()

trackcam(cam, m)

serflag = False

if mythread2.isAlive():
    print 'mythread2 is alive'
if listenthread.isAlive():
    print 'listenthread is alive'
del (cam)
ser.close()
del ser
ser_sp1.close()
del ser_sp1
ser_sp2.close()
del ser_sp2
ser_p1.close()
del ser_p1
sys.exit()
```

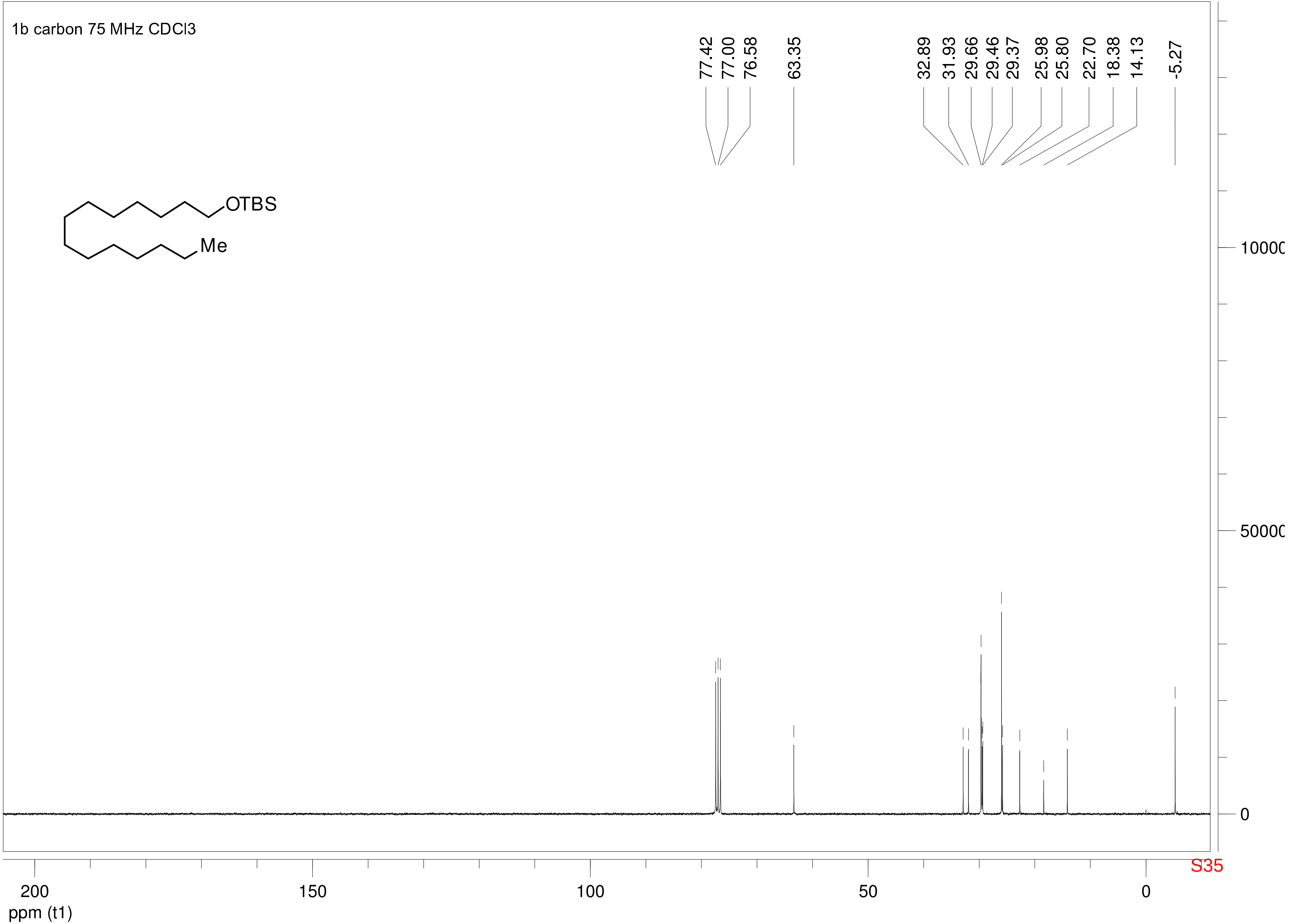
```
#RPI.py this script runs on the Raspberry Pi computer

import time
import serial
import threading
from Queue import Queue
import RPi.GPIO as gpio

import os

gpio.setmode (gpio.BOARD)

#each A4988 motor driver has 5 connections to the Rpi:
#step: a pulse makes the motor move one step
#dir: sets the direction of the motor
#ms1 ,ms2, ms3: microstepping
#for all motors, full microstepping was always on
#(ms1, ms2, ms3 all connected to + 3.3V).

#the connections to the motors are set to outputs on the rpi:
xstep = 38
xdir = 40

ystep = 16
ydir = 18

zstep = 33
zdir = 35

mstep = 37 #a single output (high) sets all motors to full microstepping

twcstep = 32
twcdir = 36

twbstep = 7
twbdir = 11

twastep = 13
twadir = 15

tpostep = 29
tpodir = 31

gpio.setup(mstep, gpio.OUT)
gpio.setup(xstep, gpio.OUT)
gpio.setup(xdir, gpio.OUT)
gpio.setup(ystep, gpio.OUT)
gpio.setup(ydir, gpio.OUT)
gpio.setup(zstep, gpio.OUT)
gpio.setup(zdir, gpio.OUT)
gpio.setup(twastep, gpio.OUT)
gpio.setup(twbstep, gpio.OUT)
gpio.setup(twcstep, gpio.OUT)
gpio.setup(twadir, gpio.OUT)
gpio.setup(twbdir, gpio.OUT)
gpio.setup(twcdir, gpio.OUT)
gpio.setup(tpostep, gpio.OUT)
gpio.setup(tpodir, gpio.OUT)


gpio.output(xstep, False)
gpio.output(ystep, False)
gpio.output(zstep, False)
gpio.output(xdir, False)
gpio.output(ydir, False)
gpio.output(zdir, False)
gpio.output(twadir, False)
gpio.output(twbdir, False)
gpio.output(twcdir, False)
gpio.output(twastep, False)
gpio.output(twbstep, False)
gpio.output(twcstep, False)
gpio.output(tpostep, False)
gpio.output(tpodir, False)

gpio.output(mstep, True)


#motors:
#x, y, z, three 3-way valves, tap-outlet. 7 in total.
#7 functions/threads/queues.
#mvx, mvy, mvz, twa, twb, twc, tpo

#for rpi:
ser = serial.Serial('/dev/ttyAMA0', 9600, timeout = 0.01)

flag = True

xq = Queue(maxsize=0) #for x motor
yq = Queue(maxsize=0) #for y motor
zq = Queue(maxsize=0) #for z motor
aq = Queue(maxsize=0) #for 3-way-valve 1 motor
bq = Queue(maxsize=0) #for 3-way-valve 2 motor
cq = Queue(maxsize=0) #for 3-way-valve 3 motor
tpoq = Queue(maxsize=0) #for aqueous-out 2-way-valve motor

#movex, movey, movez, all run in separate threads
#take next command from each queue and move motor accordingly

def movex():
    global xq
    while flag:
        a = xq.get()
        print ('mvx ' + str(a)).strip()
    print ('yes')
        if int(a)<0:
            gpio.output(xdir, False)
        if int(a)>0:
            gpio.output(xdir, True)
        for x in range(abs(int(a))):
            gpio.output(xstep, True)
            time.sleep(0.001)
            gpio.output(xstep, False)
            time.sleep(0.001)


def movey():
    global yq
    while flag:
        a = yq.get()
        print ('mvy ' + str(a)).strip()
        if int(a)<0:
            gpio.output(ydir, False)
        if int(a)>0:
            gpio.output(ydir, True)
        for x in range(abs(int(a))):
            gpio.output(ystep, True)
            time.sleep(0.001)
            gpio.output(ystep, False)
            time.sleep(0.001)


def movez():
    global zq
    while flag:
        a = zq.get()
        print ('mvz ' + str(a)).strip()
        if int(a)<0:
            gpio.output(zdir, False)
        if int(a)>0:
            gpio.output(zdir, True)
        for x in range(abs(int(a))):
            gpio.output(zstep, True)
            time.sleep(0.001)
            gpio.output(zstep, False)
            time.sleep(0.001)

#turna, turnb, turnc for each of the 3-way valves:

def turna():
    global aq
    while flag:
        a = aq.get()
        print ('twa ' + str(a)).strip()
        if int(a)<0:
            gpio.output(twadir, False)
        if int(a)>0:
            gpio.output(twadir, True)
        for x in range(abs(int(a))):
            gpio.output(twastep, True)
            time.sleep(0.001)
            gpio.output(twastep, False)
            time.sleep(0.001)



def turnb():
    global bq
    while flag:
        a = bq.get()
        print ('twb ' + str(a)).strip()
        if int(a)<0:
            gpio.output(twbdir, False)
        if int(a)>0:
            gpio.output(twbdir, True)
        for x in range(abs(int(a))):
            gpio.output(twbstep, True)
            time.sleep(0.001)
            gpio.output(twbstep, False)
            time.sleep(0.001)

def turnc():
    global cq
    while flag:
        a = cq.get()
        print ('twc ' + str(a)).strip()
        if int(a)<0:
            gpio.output(twcdir, False)
        if int(a)>0:
            gpio.output(twcdir, True)
        for x in range(abs(int(a))):
            gpio.output(twcstep, True)
            time.sleep(0.001)
            gpio.output(twcstep, False)
            time.sleep(0.001)

#turntpo for the aqueous-out 2-way valve

def turntpo():
    global tpoq
    while flag:
        a = tpoq.get()
        print ('tpo ' + str(a)).strip()
        if int(a)<0:
            gpio.output(tpodir, False)
        if int(a)>0:
            gpio.output(tpodir, True)
        for x in range(abs(int(a))):
            gpio.output(tpostep, True)
            time.sleep(0.001)
            gpio.output(tpostep, False)
            time.sleep(0.001)

def listen(): # listens to serial input, sends commands to relevant queues
#first three letters of string determing which queue the value is added to

    global xq
    global yq
    global zq
    print 'listen'
    global flag
    while flag:
        time.sleep(0.001) # reduces cpu load!
        if (ser.inWaiting()>0):
            x = ser.readline()
            #print x

            if x.rstrip() == 'x':
                flag = False

            if x.rstrip()[:3] == 'mvx':
                xq.put(x[3:])

            if x.rstrip()[:3] == 'mvy':
                yq.put(x[3:])

            if x.rstrip()[:3] == 'mvz':
                zq.put(x[3:])

            if x.rstrip()[:3] == 'twa':
                aq.put(x[3:])

            if x.rstrip()[:3] == 'twb':
                bq.put(x[3:])

            if x.rstrip()[:3] == 'twc':
                cq.put(x[3:])

            if x.rstrip()[:3] == 'tpo':
                tpoq.put(x[3:])


listenthread = threading.Thread(target = listen)
xthread = threading.Thread(target = movex)
ythread = threading.Thread(target = movey)
```

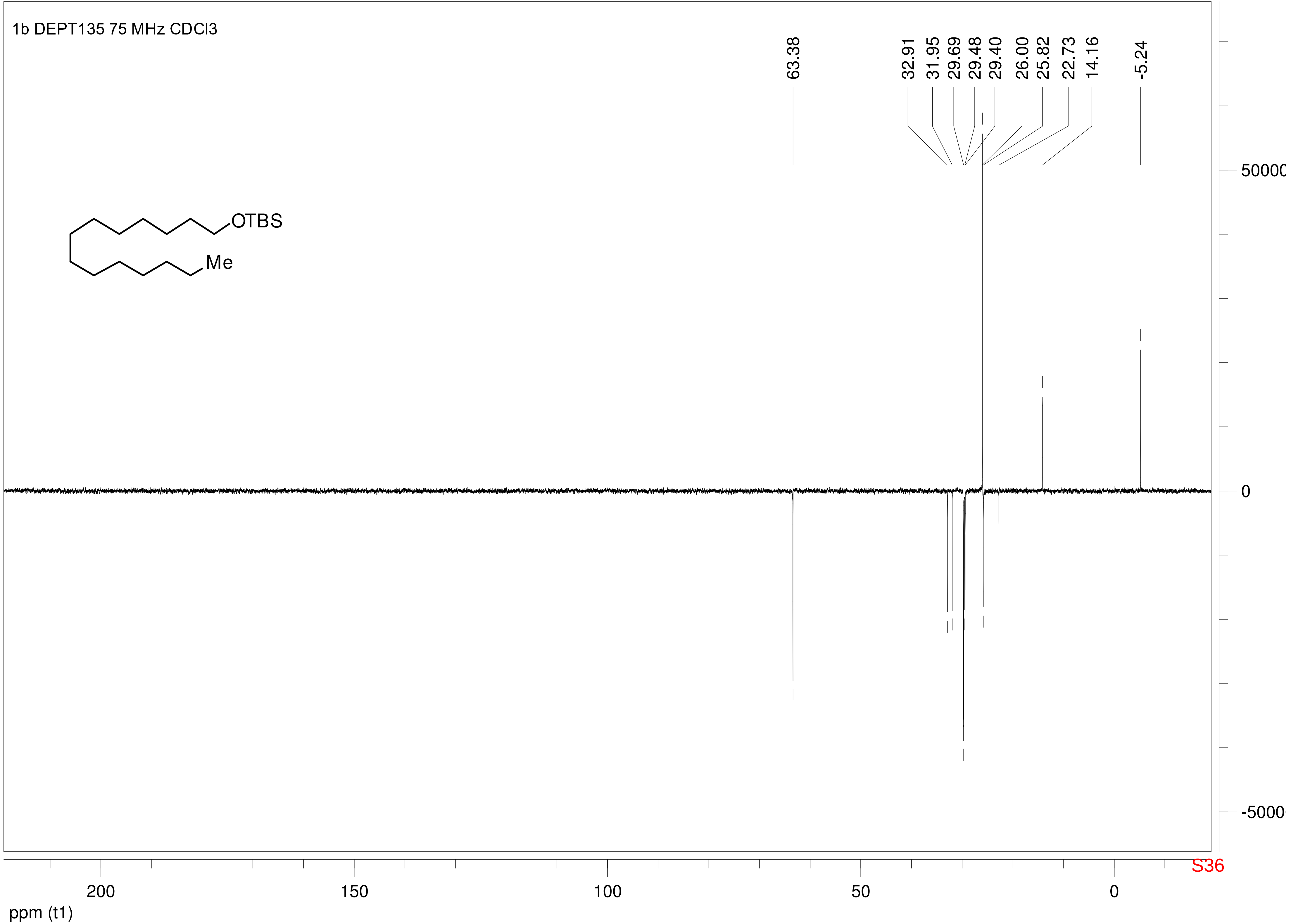
```
zthread = threading.Thread(target = movez)
athread = threading.Thread(target = turna)
bthread = threading.Thread(target = turnb)
cthread = threading.Thread(target = turnc)
tpothread = threading.Thread(target = turntpo)

xthread.start()
ythread.start()
zthread.start()
athread.start()
bthread.start()
cthread.start()
tpothread.start()

listenthread.setDaemon(True)
listenthread.start()
listenthread.join()
#movthread.join()

ser.close()
del ser
gpio.cleanup()
os._exit(0)
```


1a carbon 75 MHz CDCl3
148.56
147.32
131.86
120.91
112.42
110.98
77.42
77.00
76.58
64.70
55.87
55.73
39.15
25.91
18.31
-5.39
MeO
OMe
OTBS
50000
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)

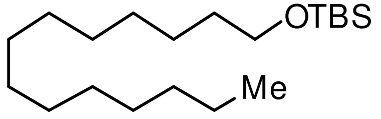

1a DEPT135 75 MHz CDCl3
120.96
112.45
111.01
64.75
55.91
55.77
39.19
25.95
-5.35
MeO
OMe
OTBS
50000
0
200
150
100
50
0
ppm (t1)

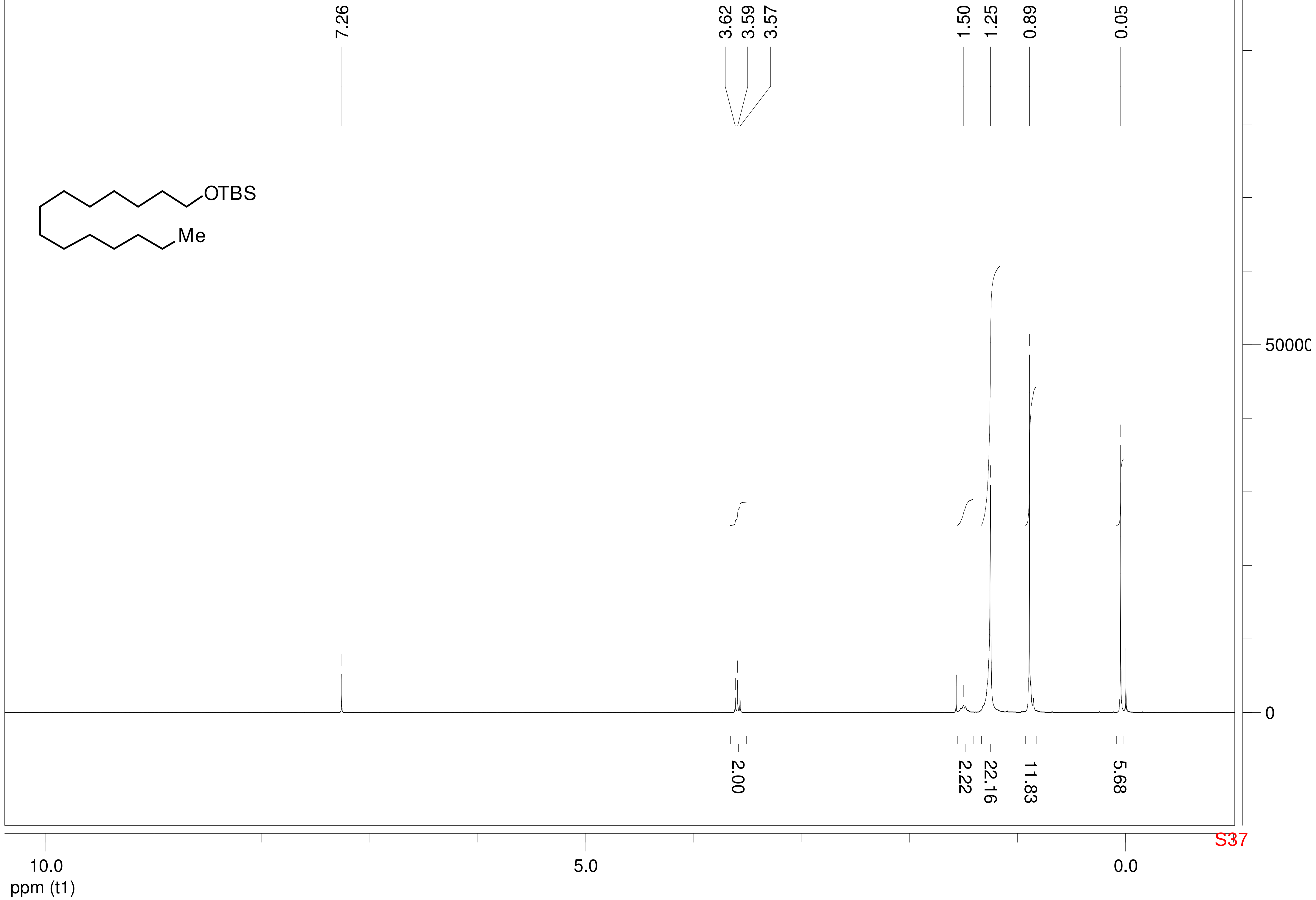

1a 300 MHz CDCl3
7.26
6.75
3.87
3.86
3.81
3.78
3.76
2.79
2.76
2.74
0.88
-0.01
MeO
OMe
OTBS
3.09
6.00
2.23
2.05
9.22
6.38
10.0
5.0
0.0
ppm (t1)
10000
50000
0

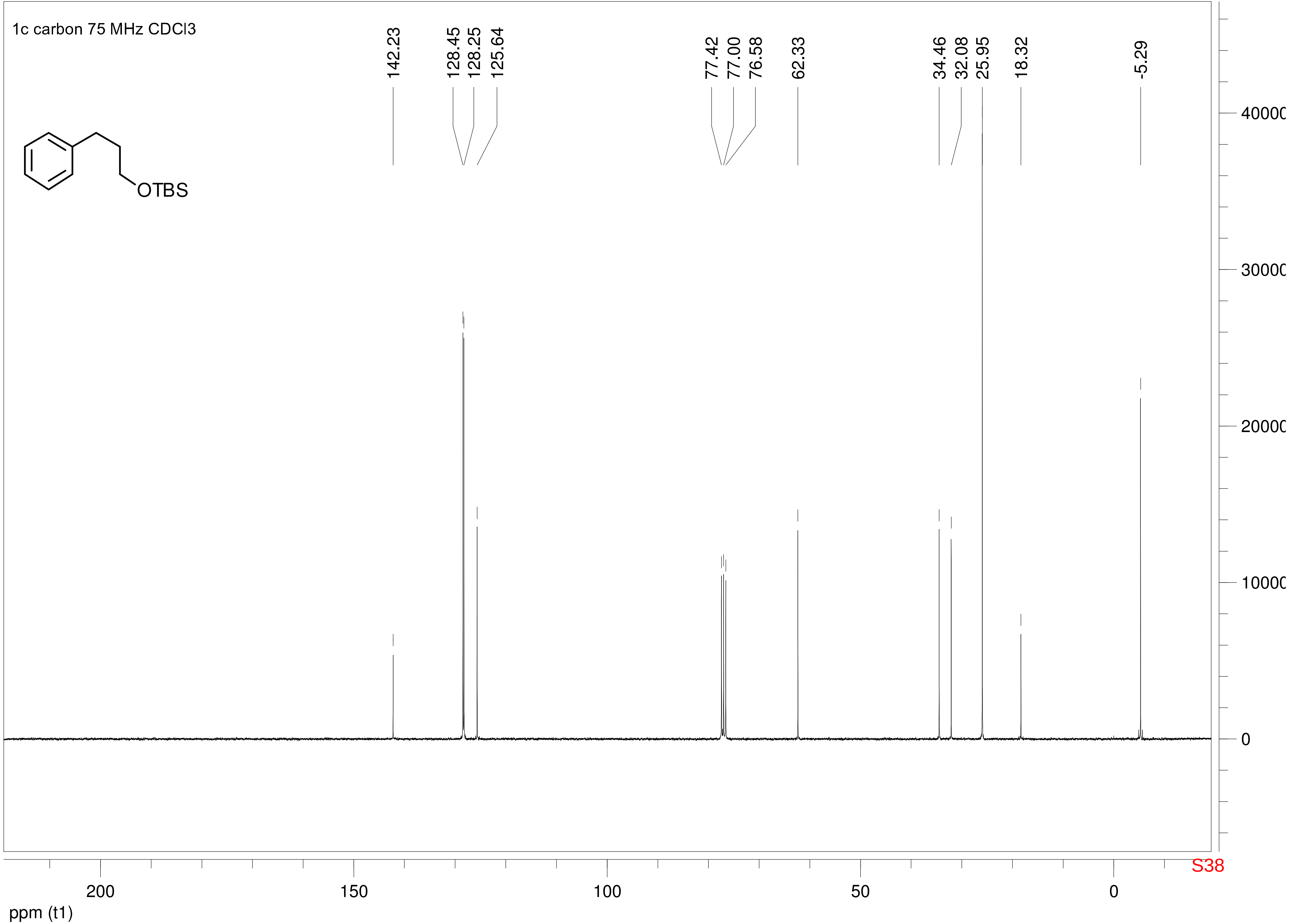

1b carbon 75 MHz CDCl3
77.42
77.00
76.58
63.35
32.89
31.93
29.66
29.46
29.37
25.98
25.80
22.70
18.38
14.13
-5.27
OTBS
Me
10000
50000
0
200
150
100
50
0
ppm (t1)


1b DEPT135 75 MHz CDCl3
63.38
32.91
31.95
29.69
29.48
29.40
26.00
25.82
22.73
14.16
-5.24
OTBS
Me
50000
0
-50000
200
150
100
50
0
ppm (t1)

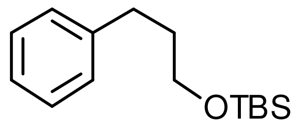
1b proton 300 MHz CDCl3

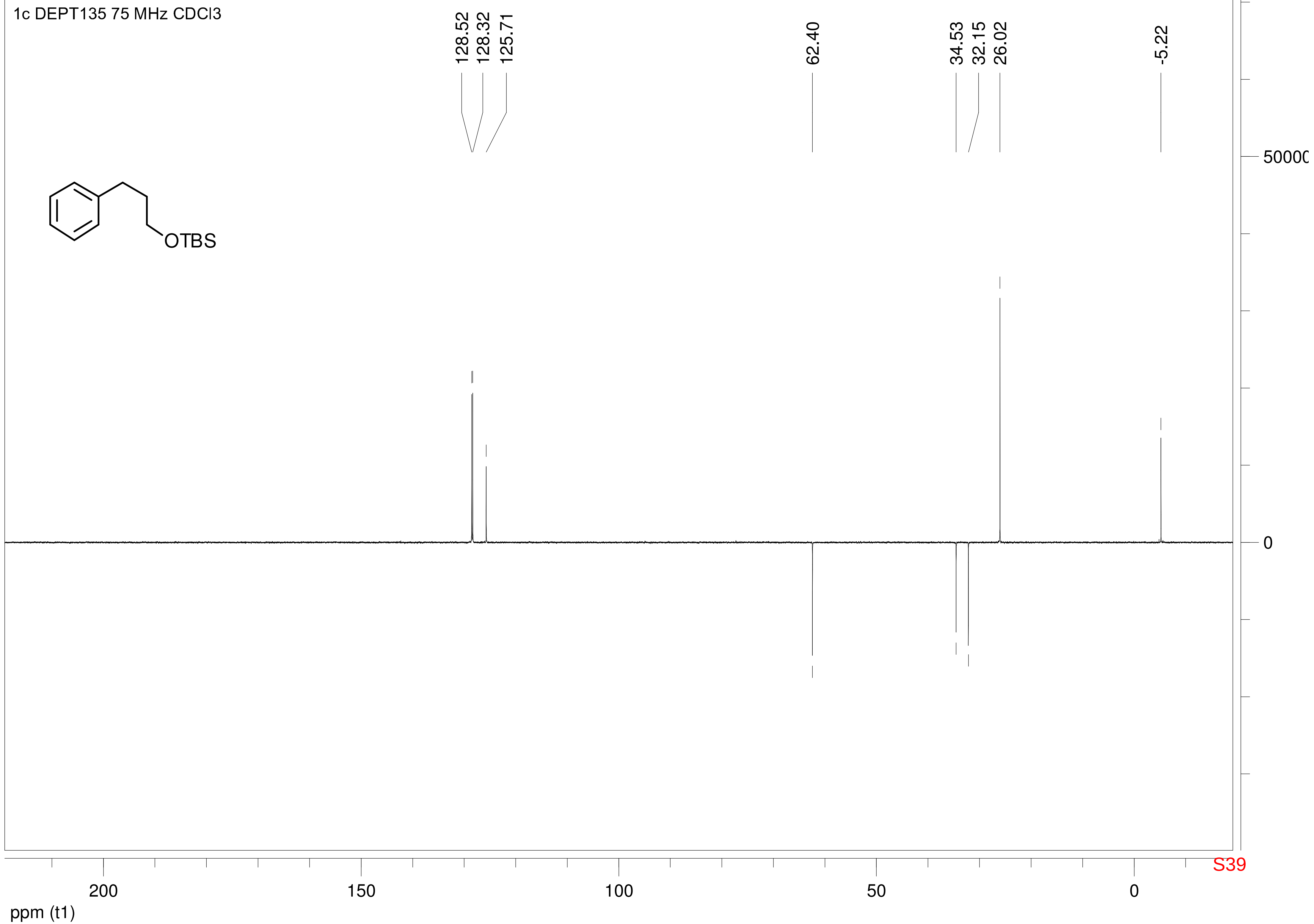




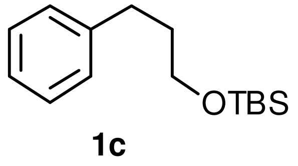

1c carbon 75 MHz CDCl3
OTBS
142.23
128.45
128.25
125.64
77.42
77.00
76.58
62.33
34.46
32.08
25.95
18.32
-5.29
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)

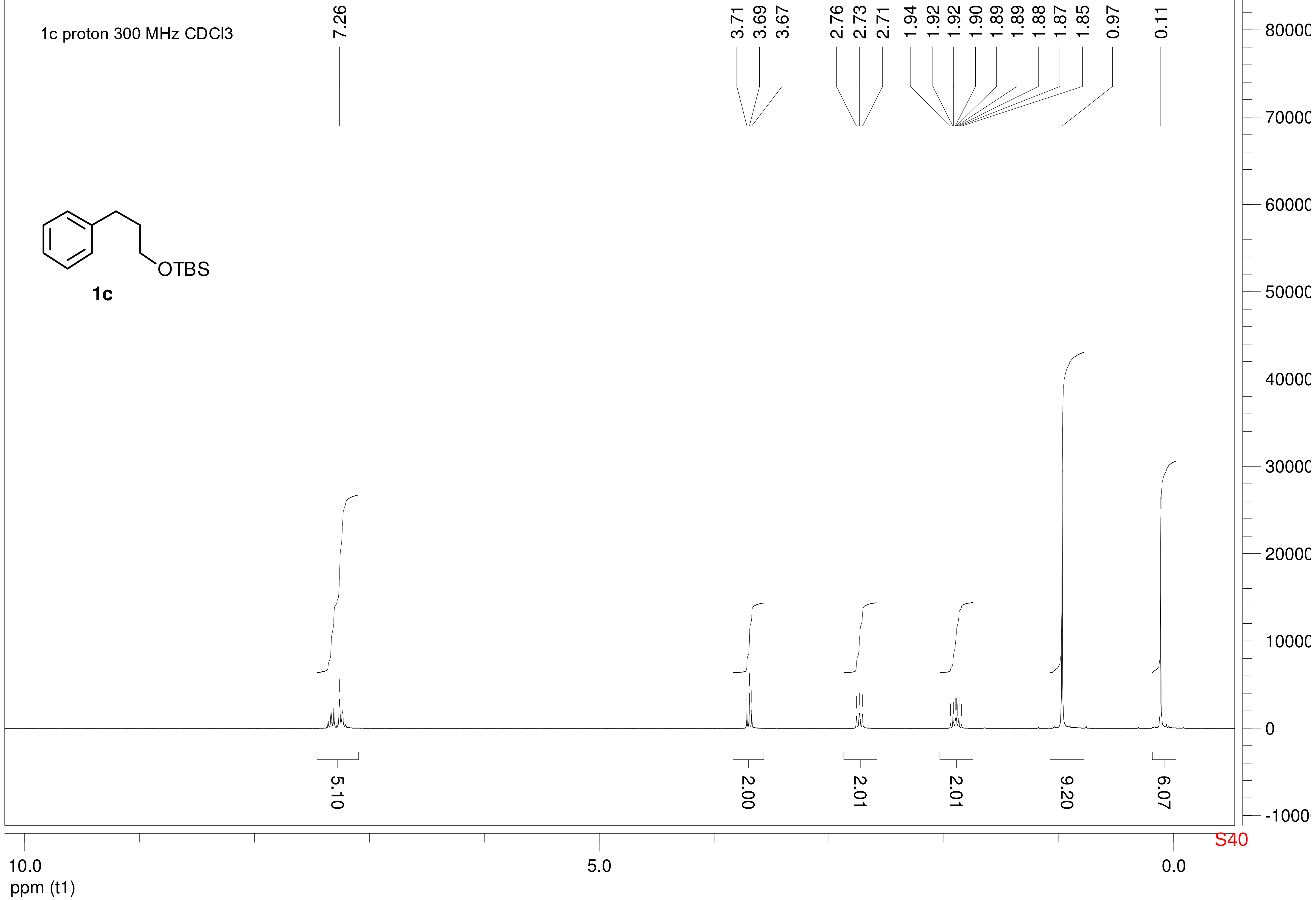
1c DEPT135 75 MHz CDCl3

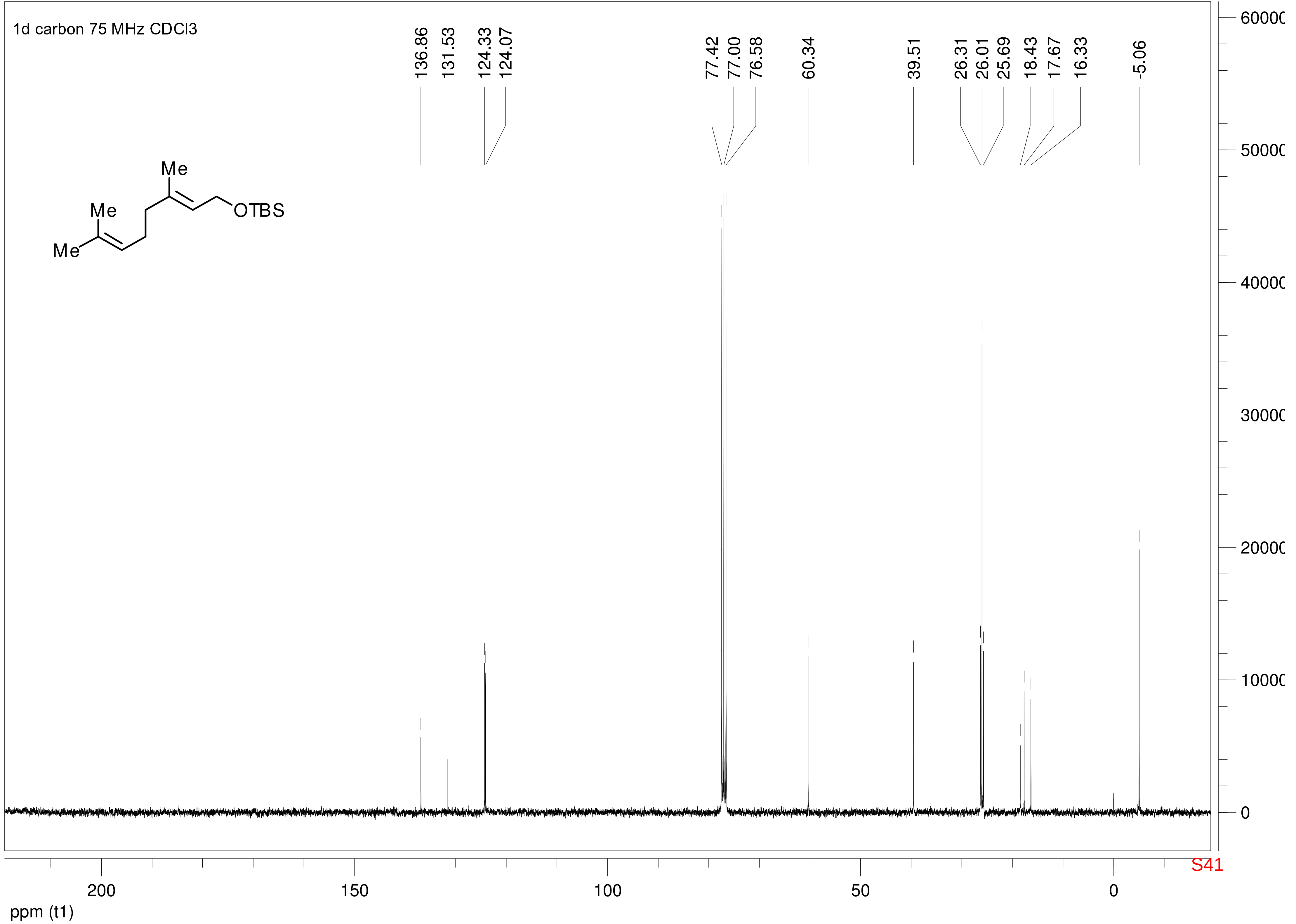


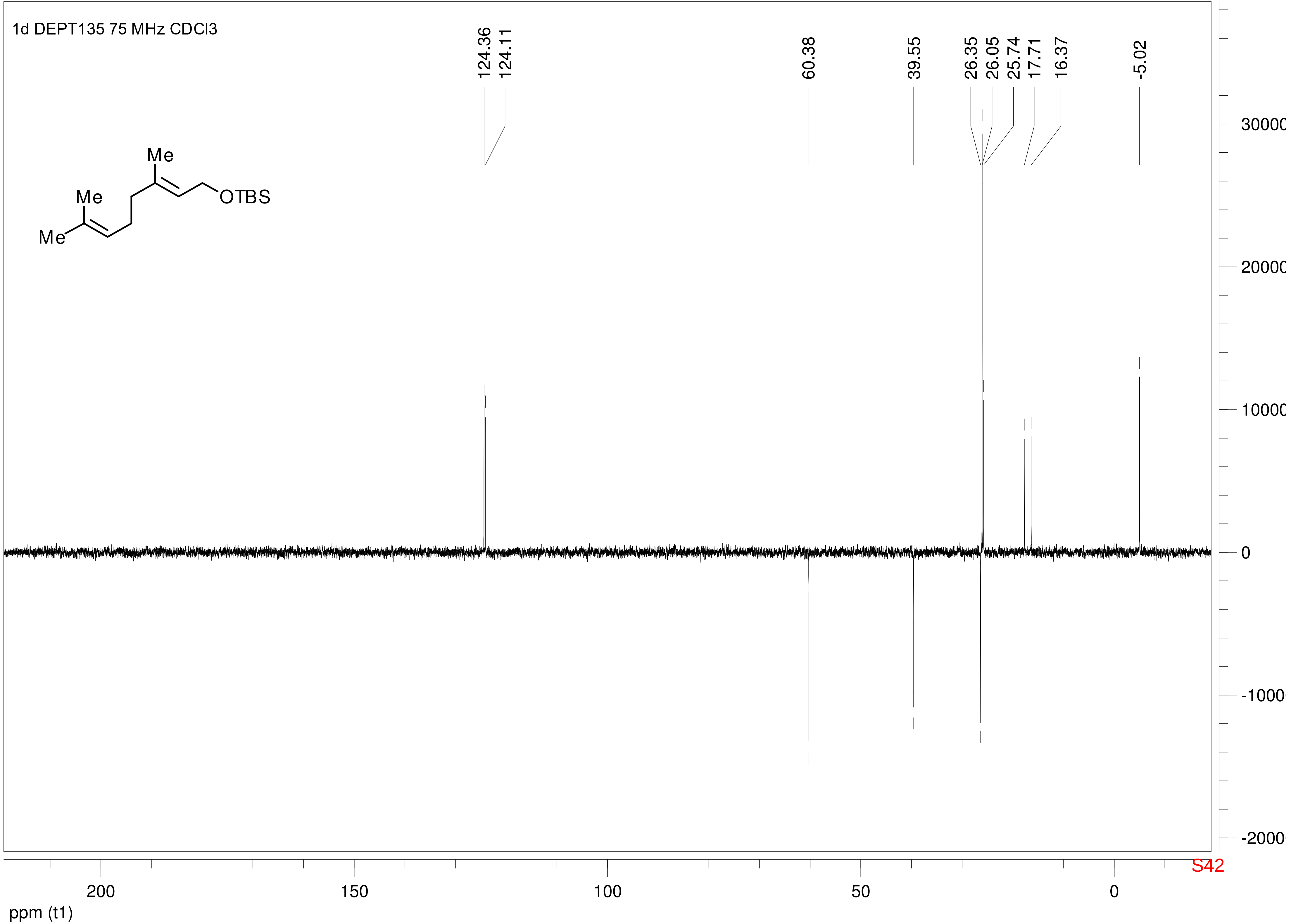


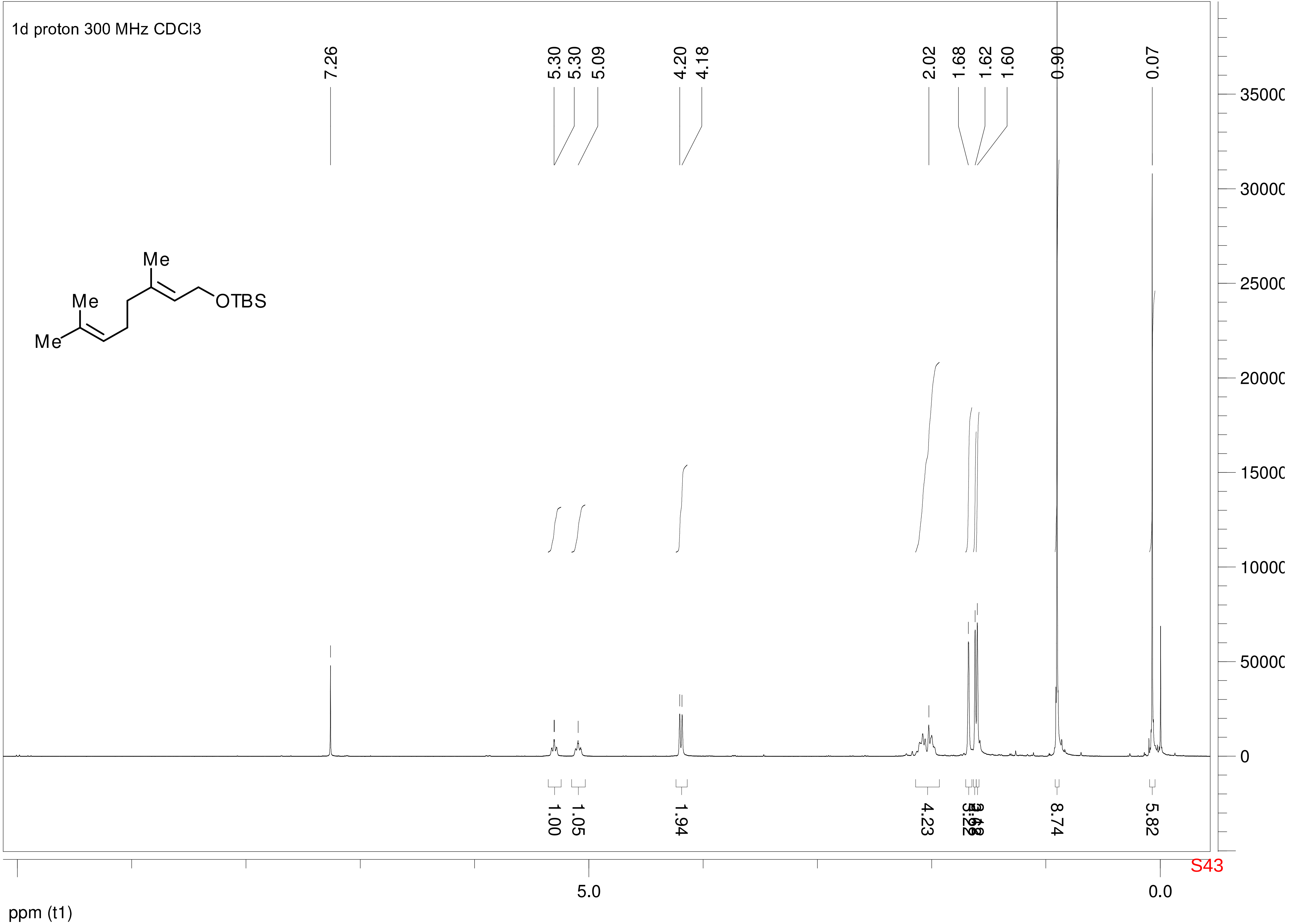
1c proton 300 MHz CDCl3



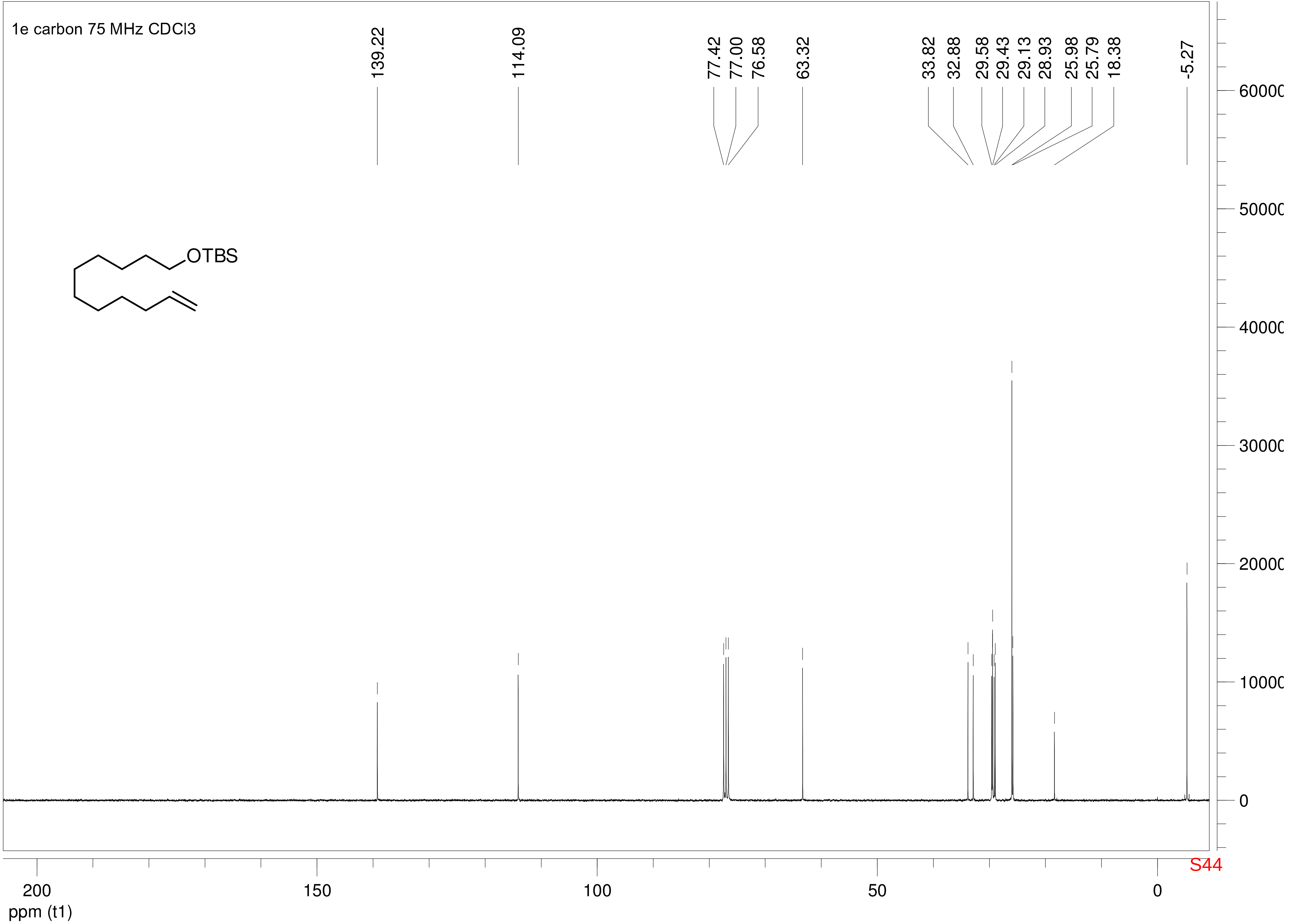



1d carbon 75 MHz CDCl3
136.86
131.53
124.33
124.07
77.42
77.00
76.58
60.34
39.51
26.31
26.01
25.69
18.43
17.67
16.33
-5.06
Me
Me
Me
OTBS
60000
50000
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)

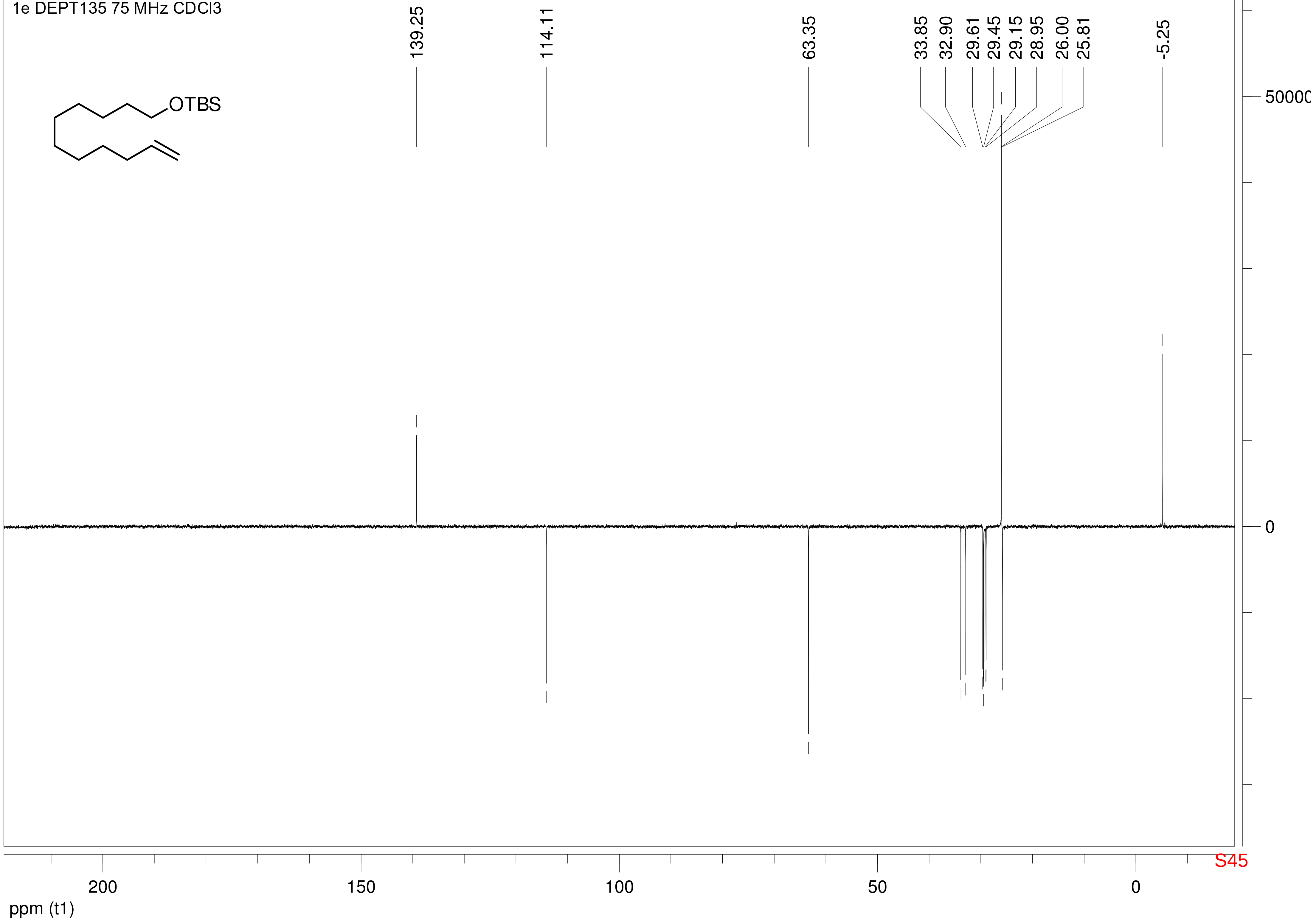

1d DEPT135 75 MHz CDCl3
124.36
124.11
60.38
39.55
26.35
26.05
25.74
17.71
16.37
-5.02
Me
Me
Me
OTBS
30000
20000
10000
0
-1000
-2000
200
150
100
50
0
ppm (t1)

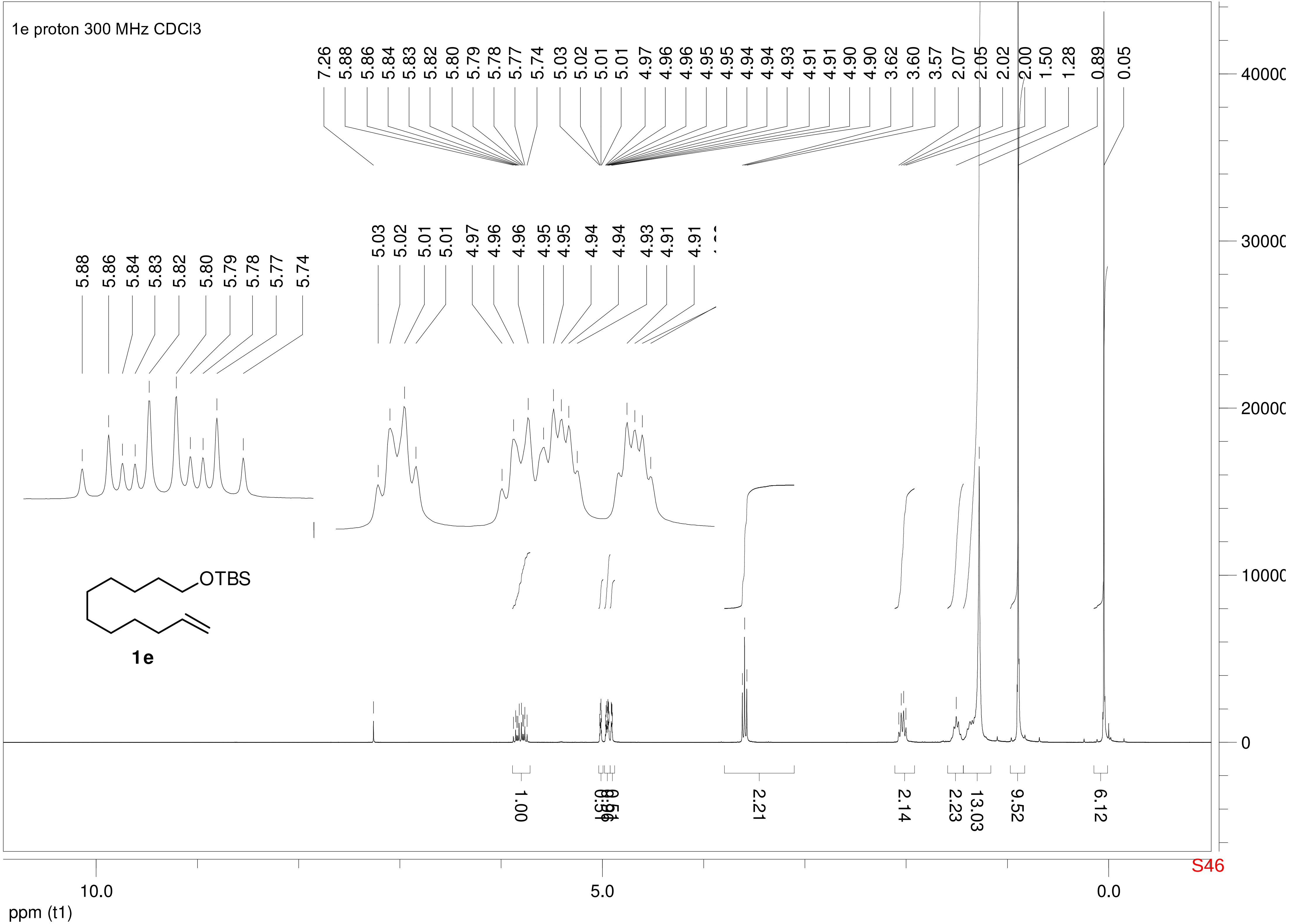


1d proton 300 MHz CDCl3
7.26
5.30
5.30
5.09
4.20
4.18
2.02
1.68
1.62
1.60
0.90
0.07
1.00
1.05
1.94
4.23
3.22
3.16
3.18
8.74
5.82
35000
30000
25000
20000
15000
10000
50000
0
5.0
0.0
ppm (t1)
Me
Me
Me
OTBS

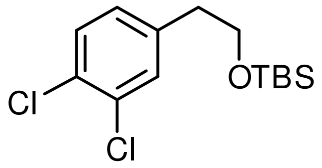


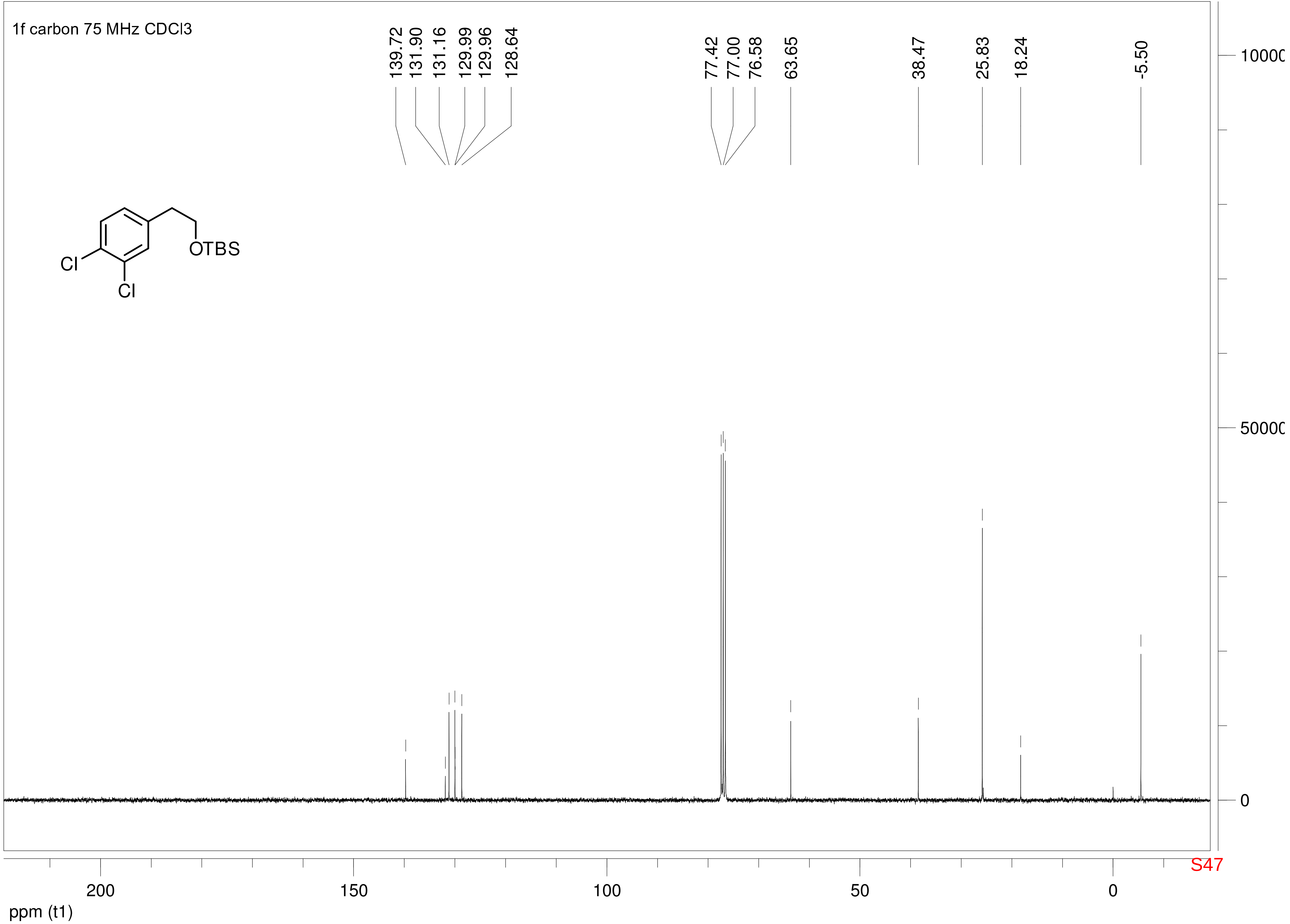
1e carbon 75 MHz CDCl3
139.22
114.09
77.42
77.00
76.58
63.32
33.82
32.88
29.58
29.43
29.13
28.93
25.98
25.79
18.38
-5.27
OTBS
60000
50000
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)

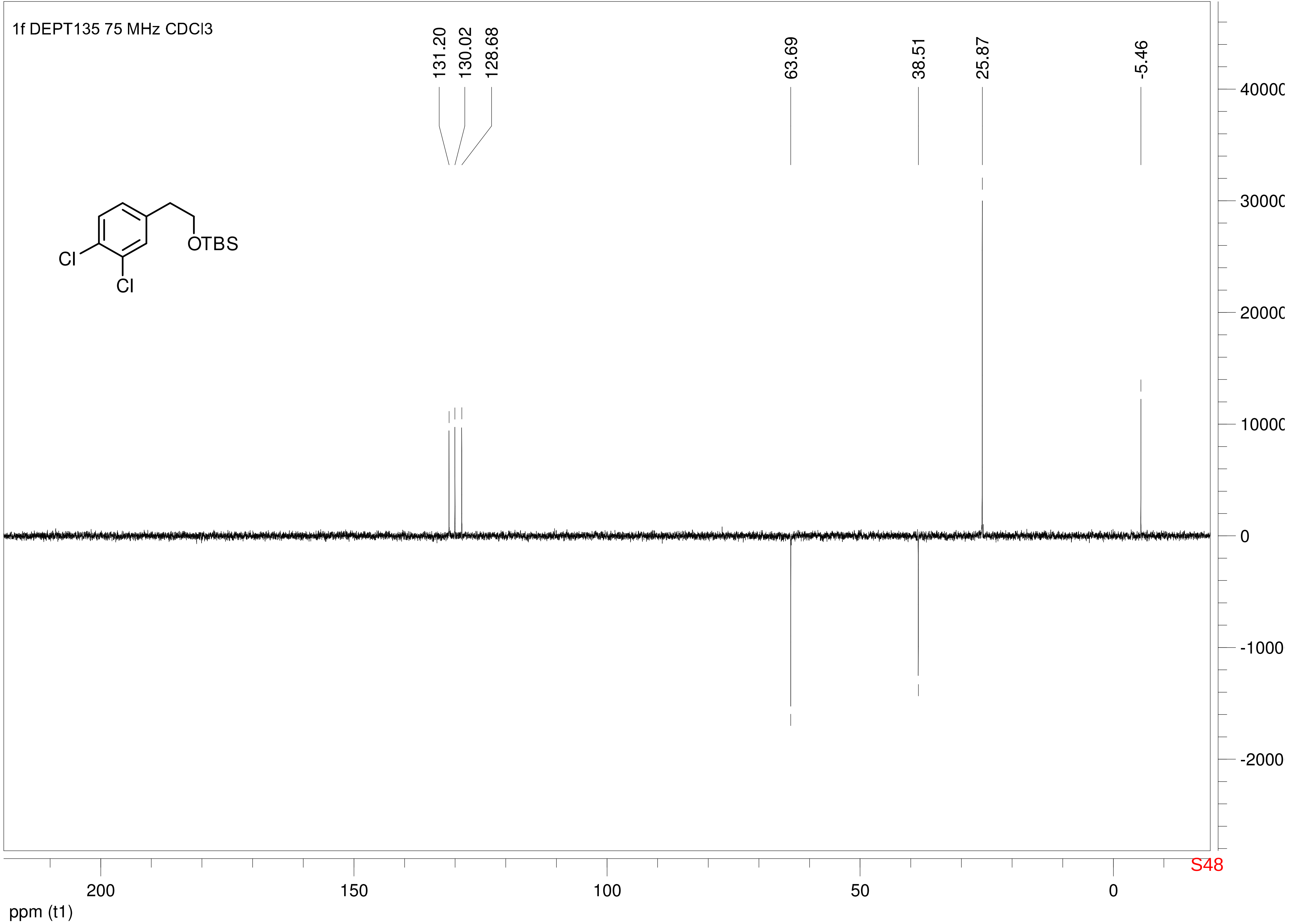

1e DEPT135 75 MHz CDCl3
OTBS
139.25
114.11
63.35
33.85
32.90
29.61
29.45
29.15
28.95
26.00
25.81
-5.25
50000
0
200
150
100
50
0
ppm (t1)

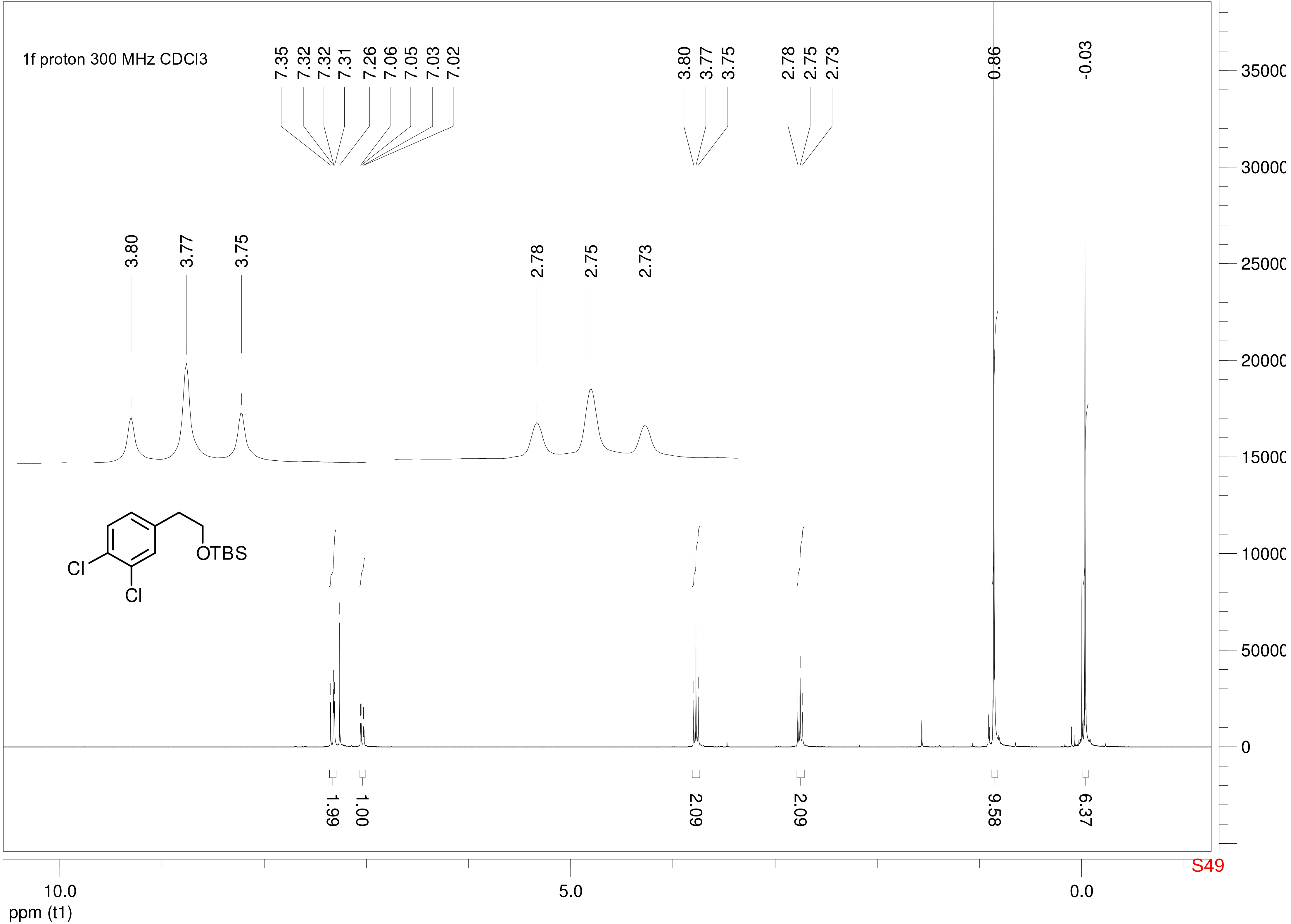

1e proton 300 MHz CDCl3
7.26
5.88
5.86
5.84
5.83
5.82
5.80
5.79
5.78
5.77
5.74
5.03
5.02
5.01
5.01
4.97
4.96
4.96
4.95
4.95
4.94
4.94
4.93
4.91
4.91
4.90
4.90
3.62
3.60
3.57
2.07
2.05
2.02
2.00
1.50
1.28
0.89
0.05
OTBS
1e
1.00
2.21
2.14
2.23
13.03
9.52
6.12
40000
30000
20000
10000
0
10.0
5.0
0.0
ppm (t1)

1f carbon 75 MHz CDCl3

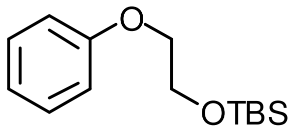


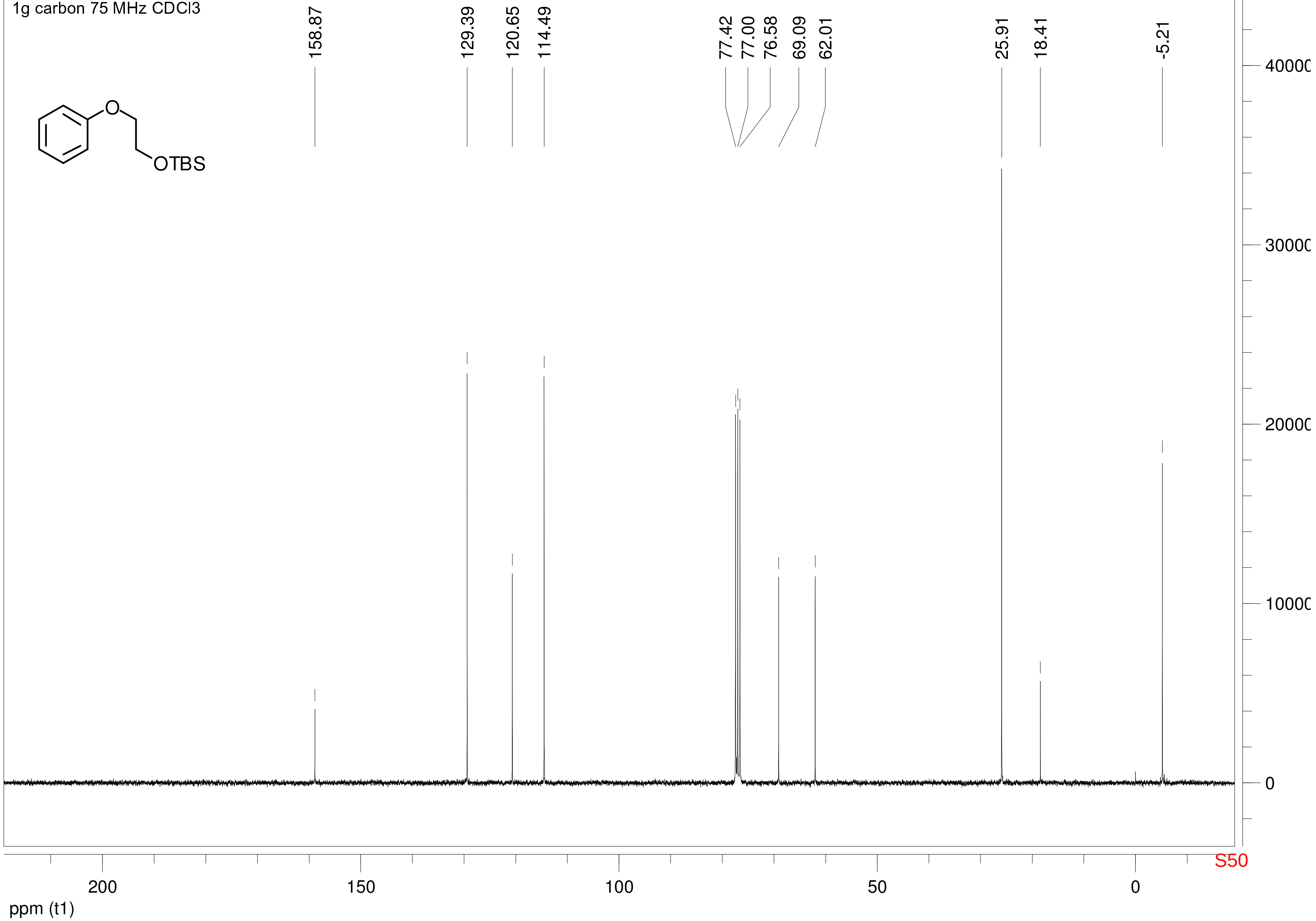


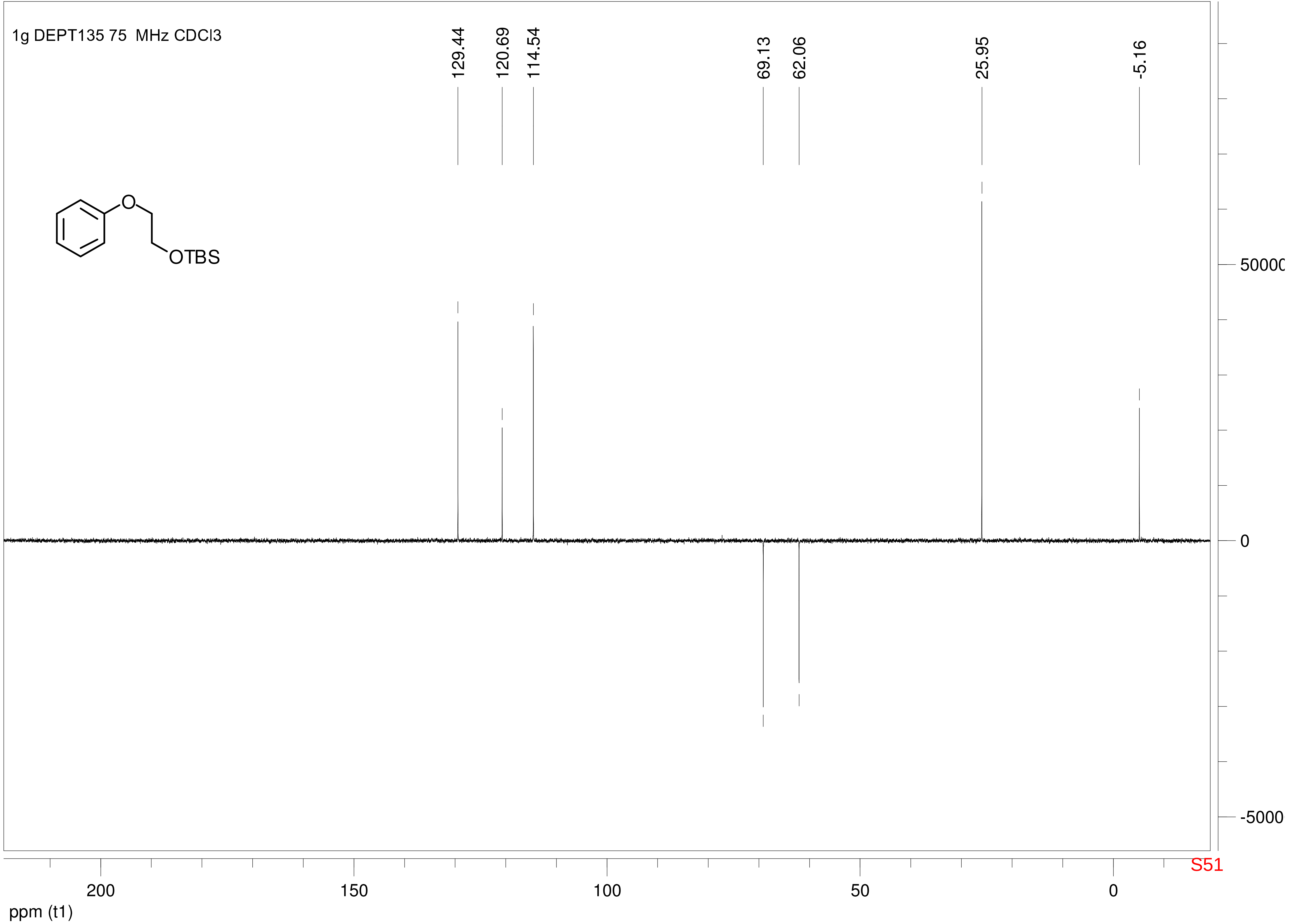

1f DEPT135 75 MHz CDCl3
131.20
130.02
128.68
63.69
38.51
25.87
-5.46
Cl
Cl
OTBS
40000
30000
20000
10000
0
-1000
-2000
200
150
100
50
0
ppm (t1)


1f proton 300 MHz CDCl3
7.35
7.32
7.32
7.31
7.26
7.06
7.05
7.03
7.02
3.80
3.77
3.75
2.78
2.75
2.73
0.86
-0.03
3.80
3.77
3.75
2.78
2.75
2.73
OTBS
Cl
Cl
1.99
1.00
2.09
2.09
9.58
6.37
35000
30000
25000
20000
15000
10000
50000
0
10.0
5.0
0.0
ppm (t1)

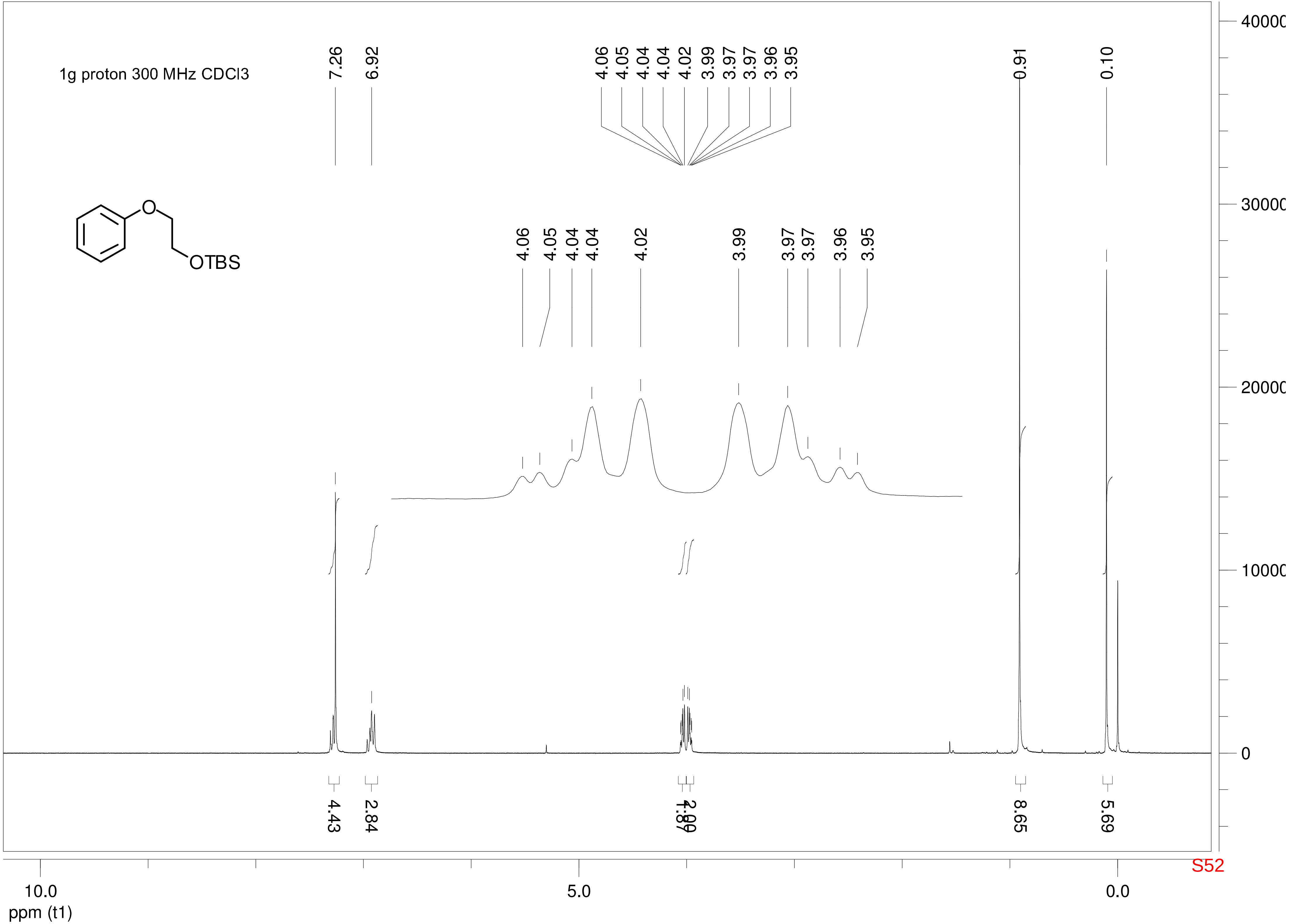
1g carbon 75 MHz CDCl3

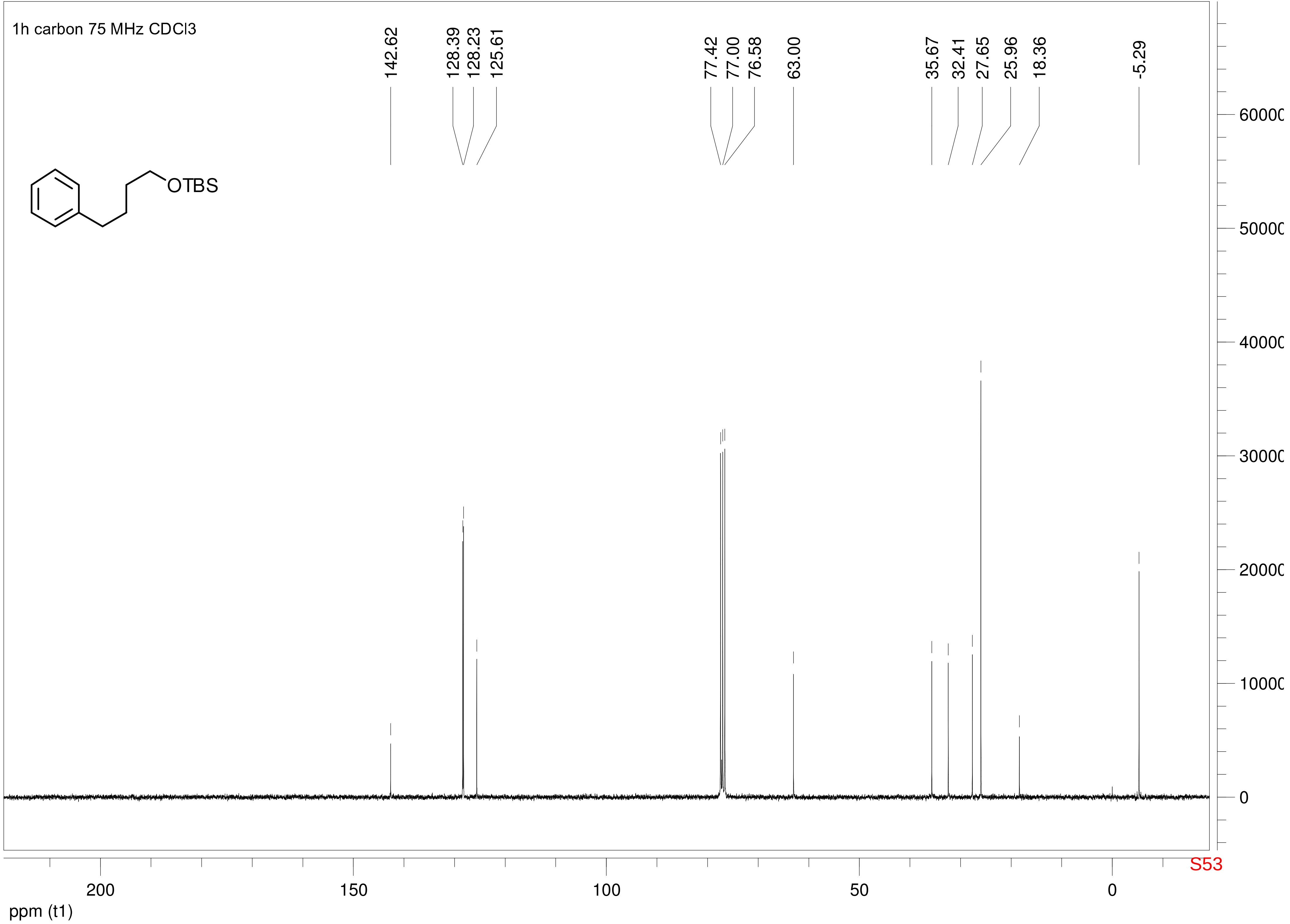


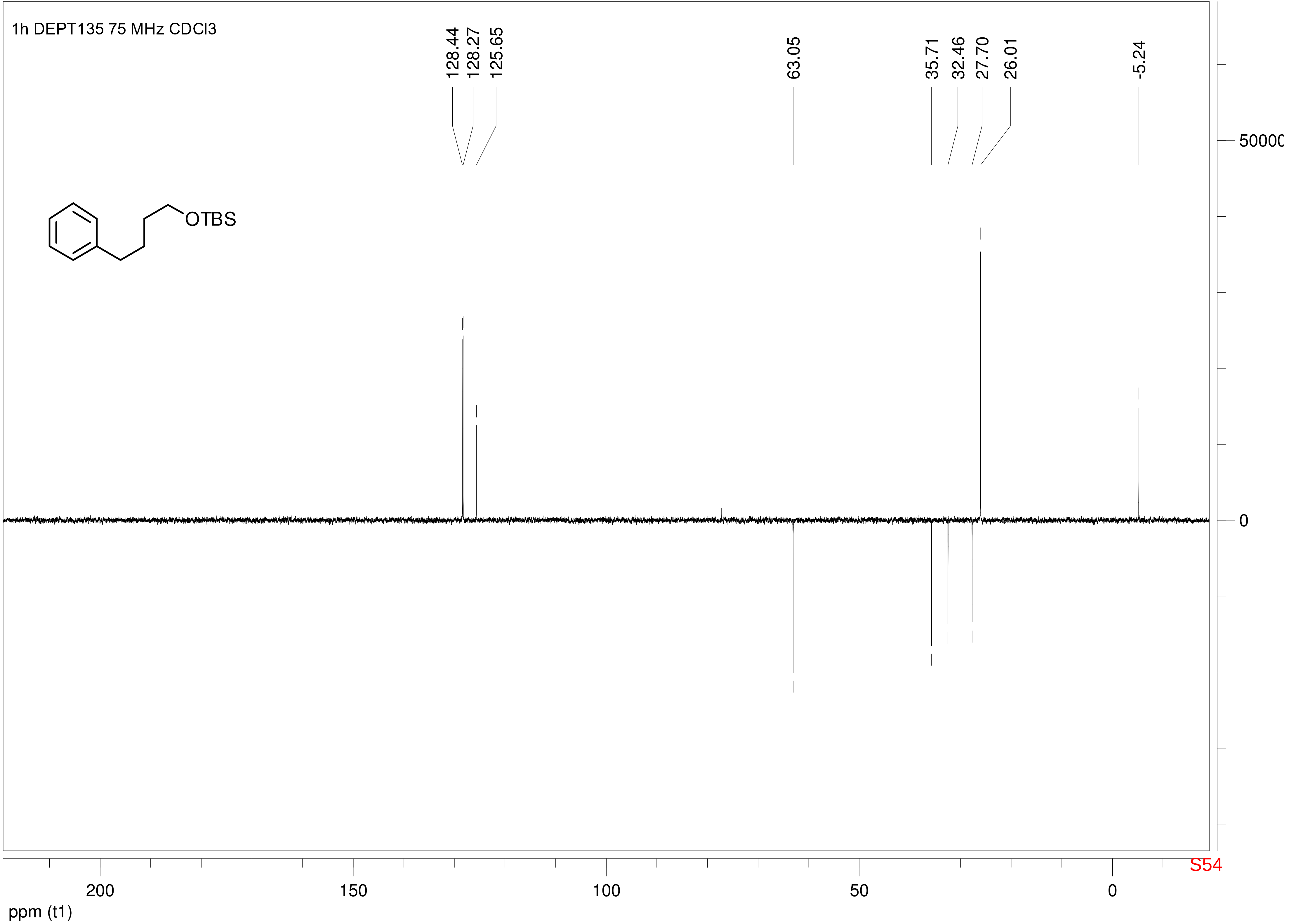


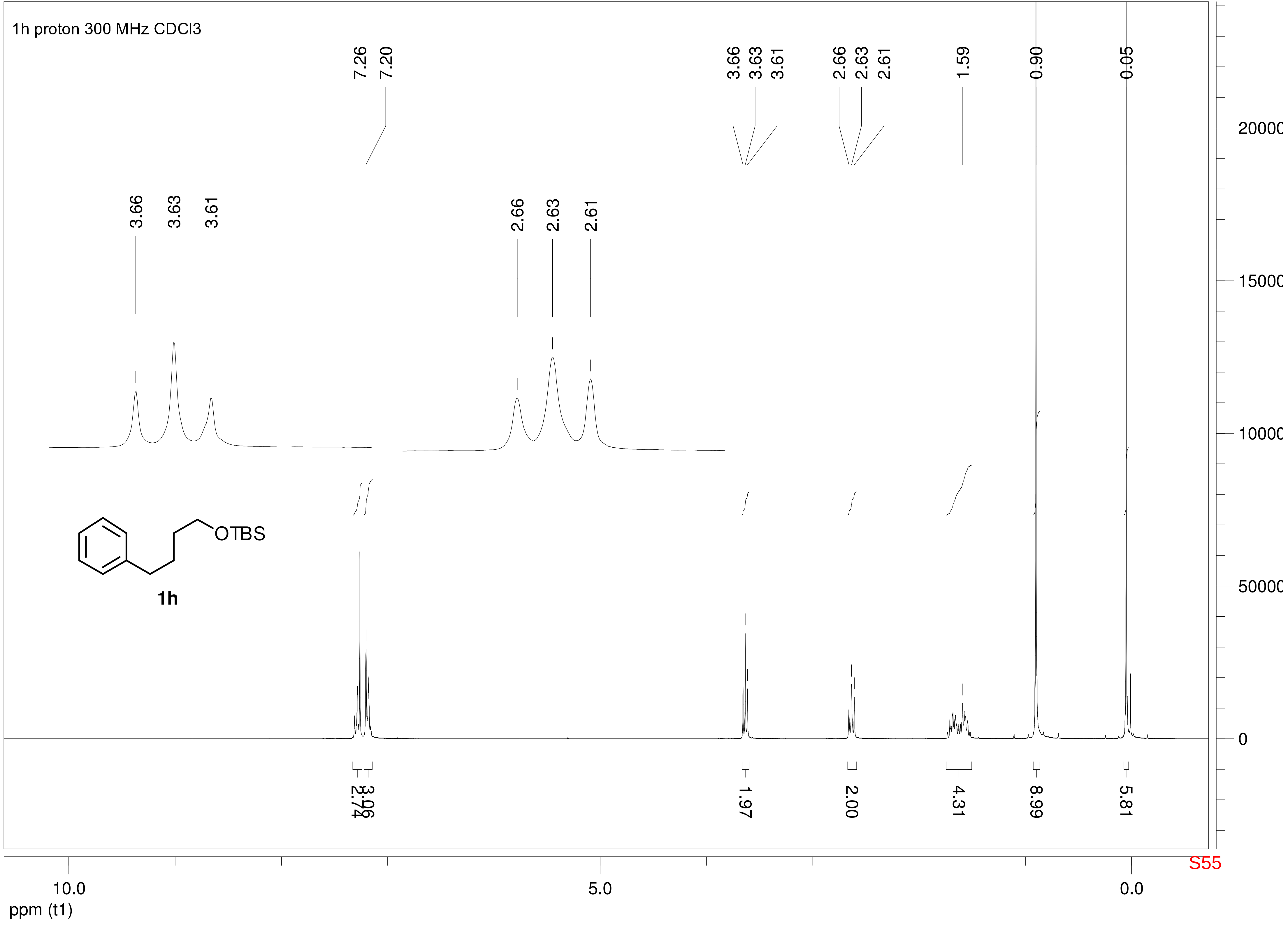


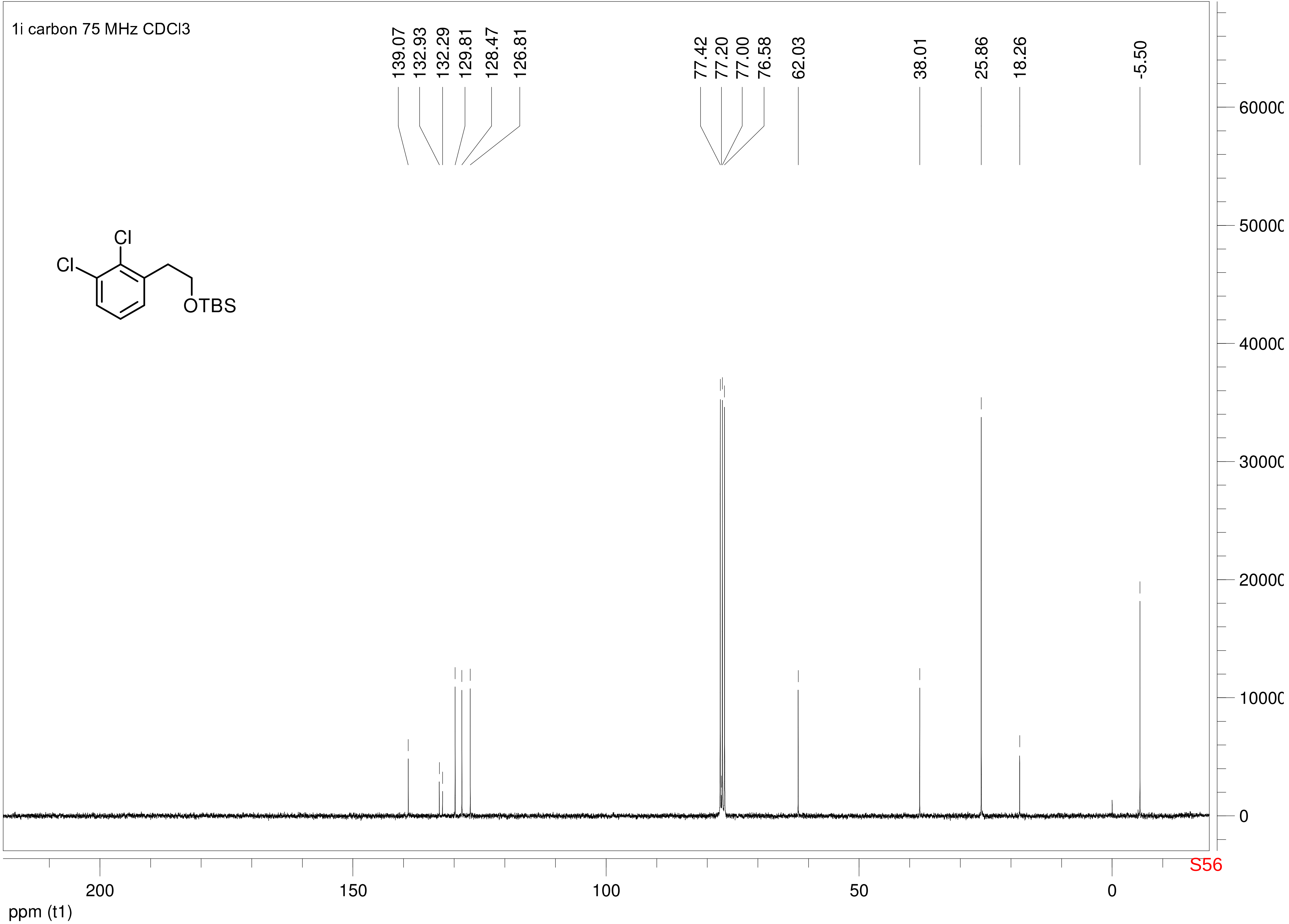
1g DEPT135 75 MHz CDCl3
129.44
120.69
114.54
69.13
62.06
25.95
-5.16
O
OTBS
50000
0
-5000
200
150
100
50
0
ppm (t1)

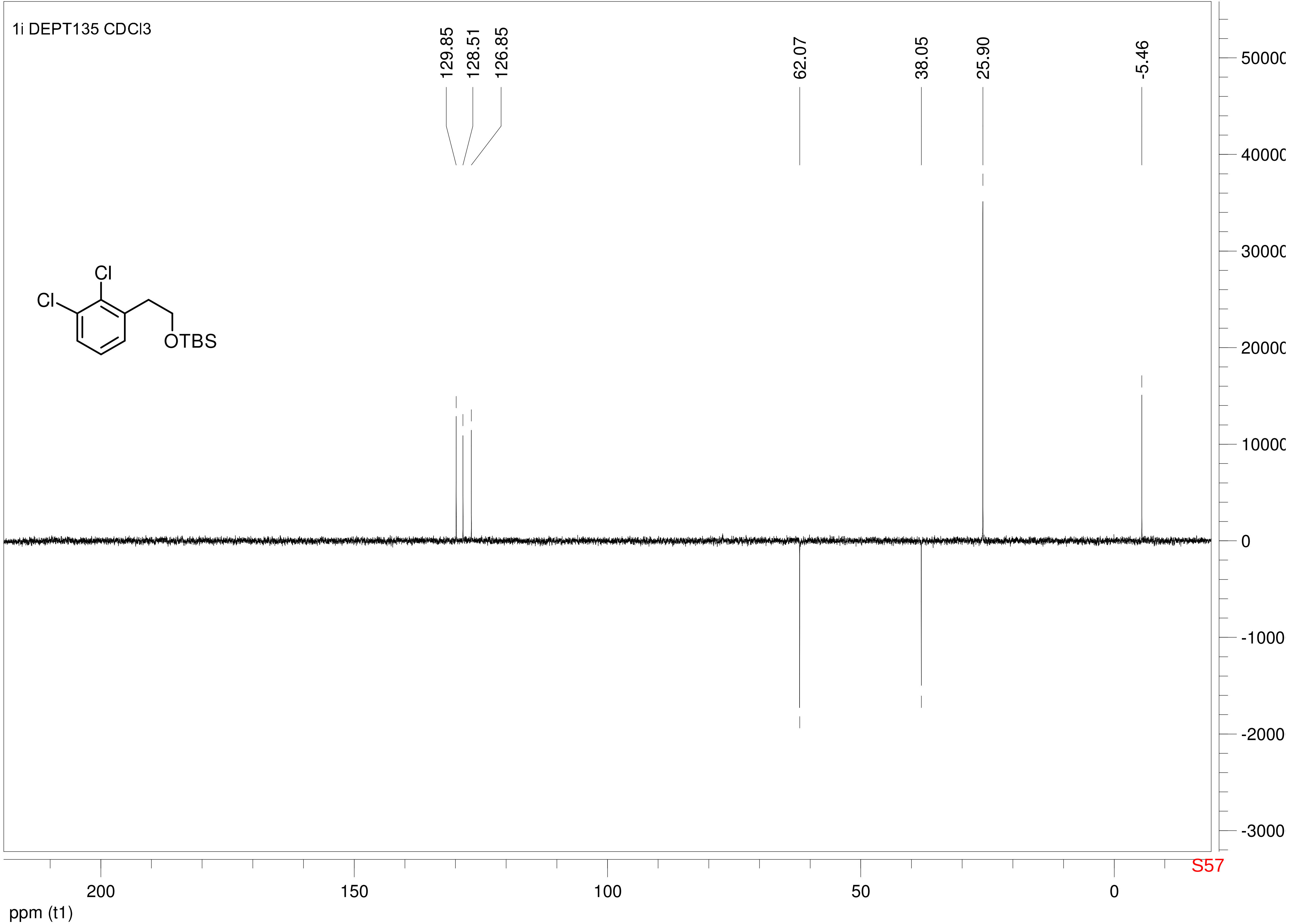

1g proton 300 MHz CDCl3
O
OTBS
7.26
6.92
4.06
4.05
4.04
4.04
4.02
3.99
3.97
3.97
3.96
3.95
0.91
0.10
4.43
2.84
1.87
2.00
8.65
5.69
40000
30000
20000
10000
0
10.0
5.0
0.0
ppm (t1)

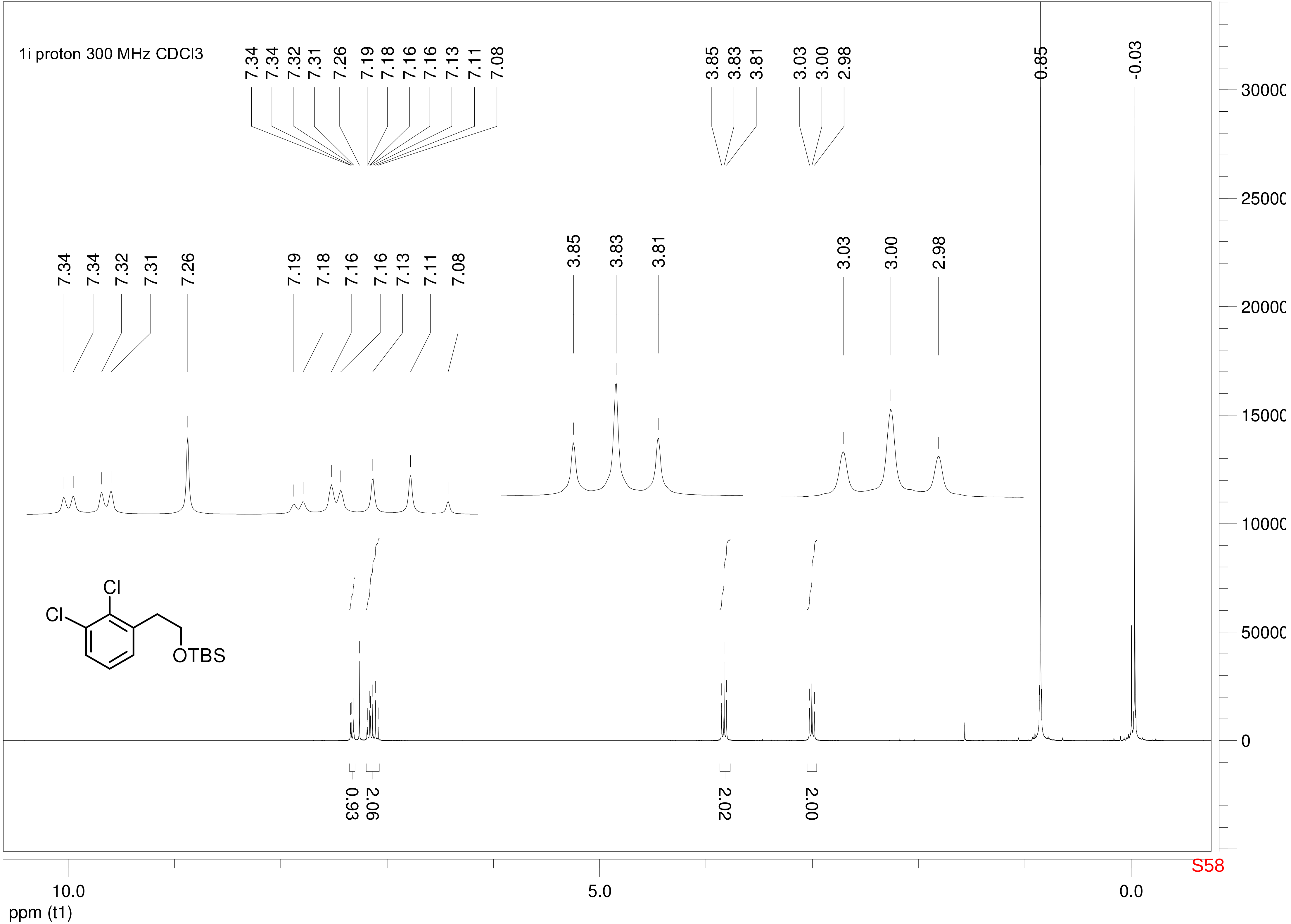

1h carbon 75 MHz CDCl3
142.62
128.39
128.23
125.61
77.42
77.00
76.58
63.00
35.67
32.41
27.65
25.96
18.36
-5.29
OTBS
60000
50000
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)


1h DEPT135 75 MHz CDCl3
128.44
128.27
125.65
63.05
35.71
32.46
27.70
26.01
-5.24
OTBS
50000
0
200
150
100
50
0
ppm (t1)


1h proton 300 MHz CDCl3
7.26
7.20
3.66
3.63
3.61
2.66
2.63
2.61
1.59
0.90
0.05
OTBS
1h
2.74
3.06
1.97
2.00
4.31
8.99
5.81
20000
15000
10000
50000
0
10.0
5.0
0.0
ppm (t1)

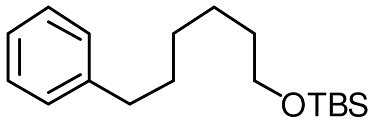

1i carbon 75 MHz CDCl3
139.07
132.93
132.29
129.81
128.47
126.81
77.42
77.20
77.00
76.58
62.03
38.01
25.86
18.26
-5.50
Cl
Cl
OTBS
60000
50000
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)

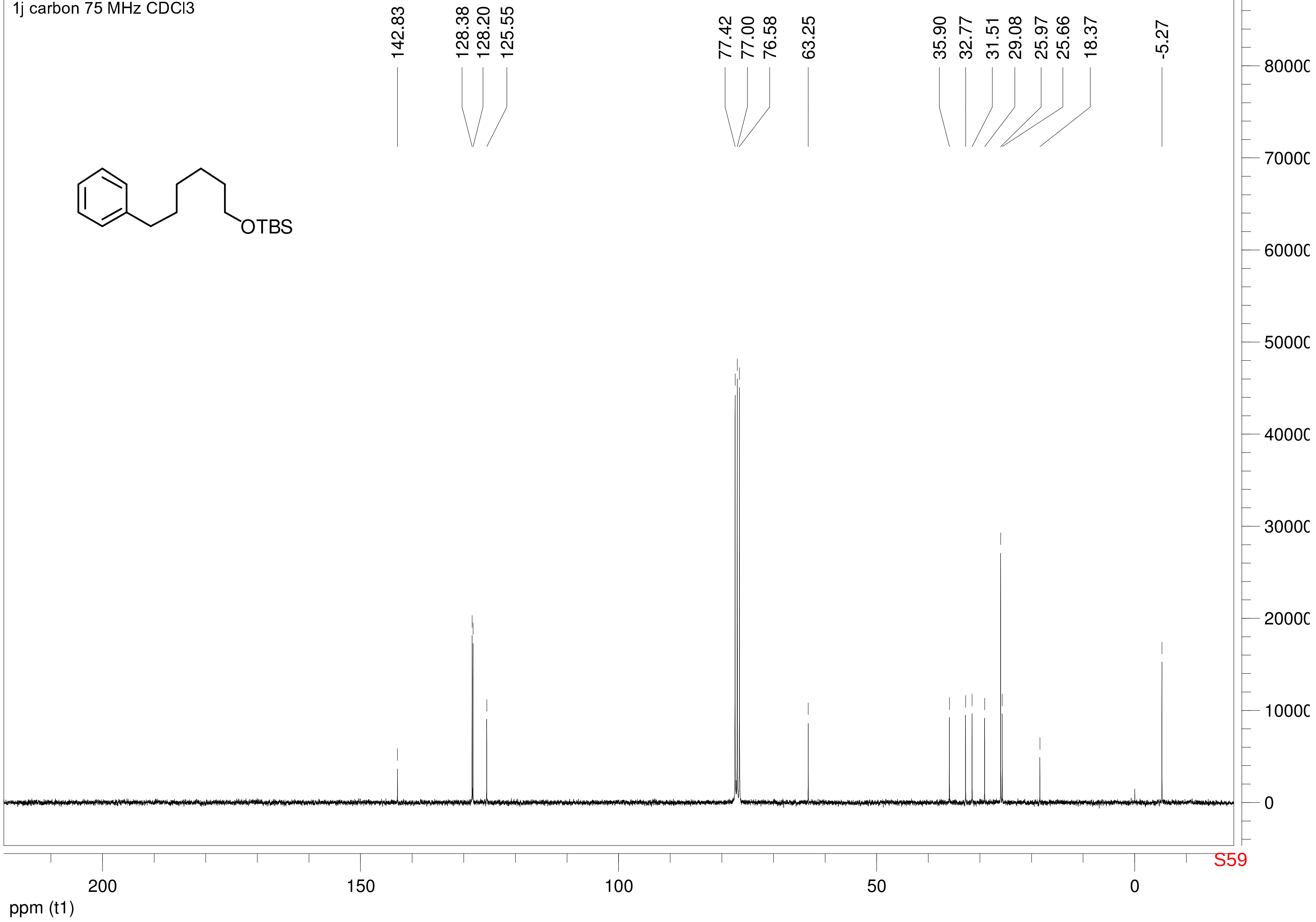

1i DEPT135 CDCl3
129.85
128.51
126.85
62.07
38.05
25.90
-5.46
Cl
Cl
OTBS
50000
40000
30000
20000
10000
0
-1000
-2000
-3000
200
150
100
50
0
ppm (t1)

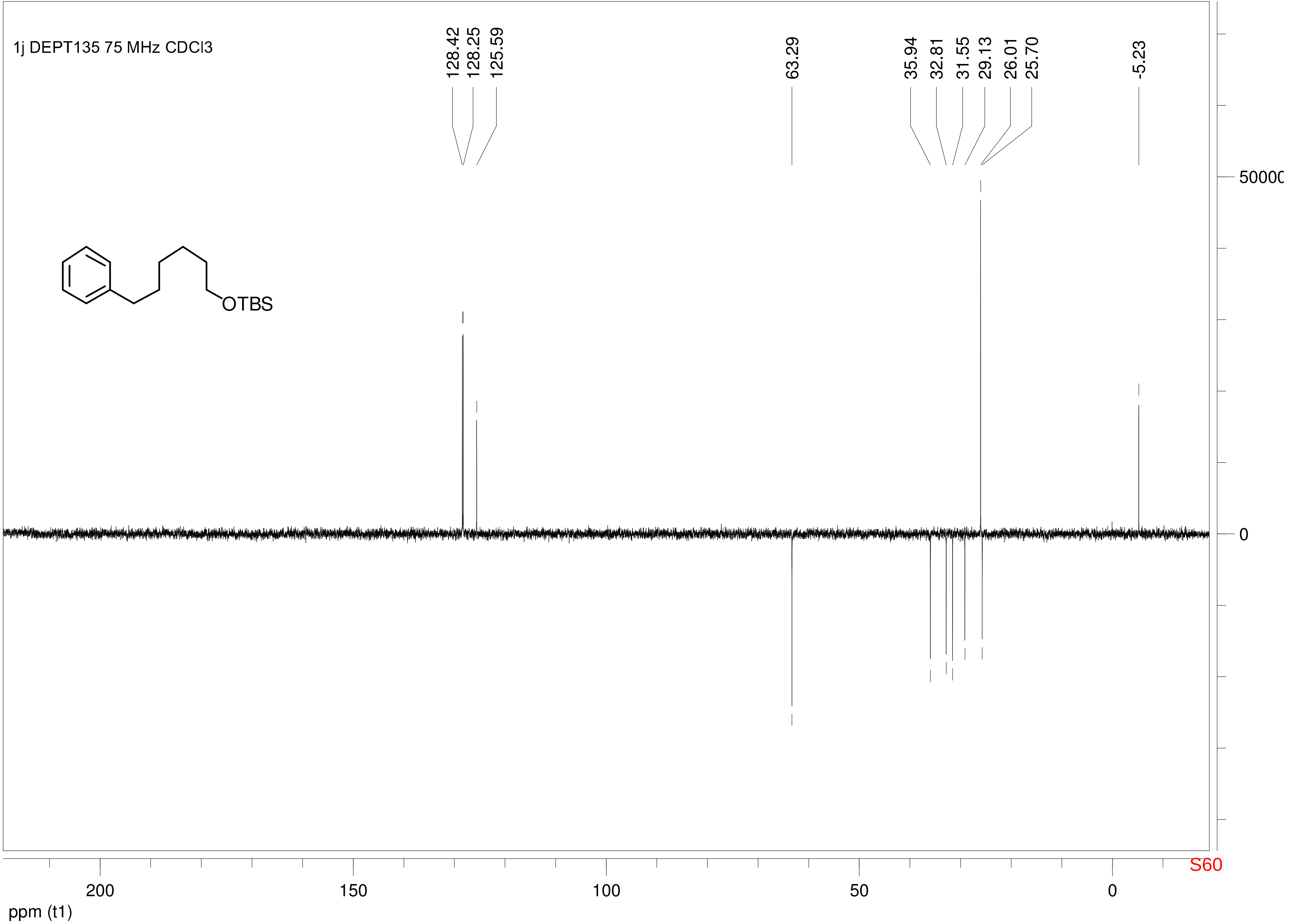


1i proton 300 MHz CDCl3
7.34
7.34
7.32
7.31
7.26
7.19
7.18
7.16
7.16
7.13
7.11
7.08
3.85
3.83
3.81
3.03
3.00
2.98
0.85
-0.03
Cl
Cl
OTBS
0.93
2.06
2.02
2.00
30000
25000
20000
15000
10000
50000
0
10.0
5.0
0.0
ppm (t1)

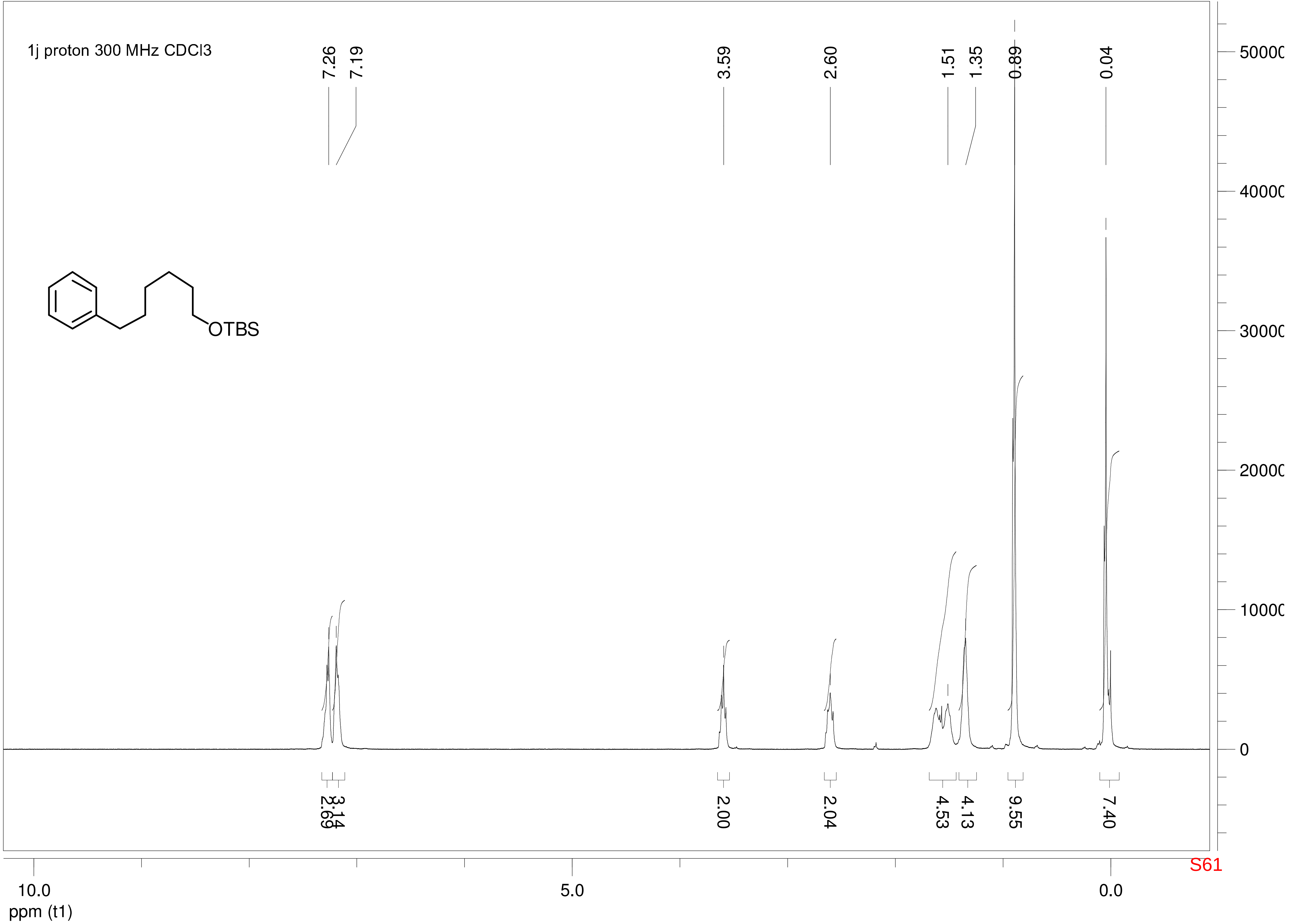
1j carbon 75 MHz CDCl3

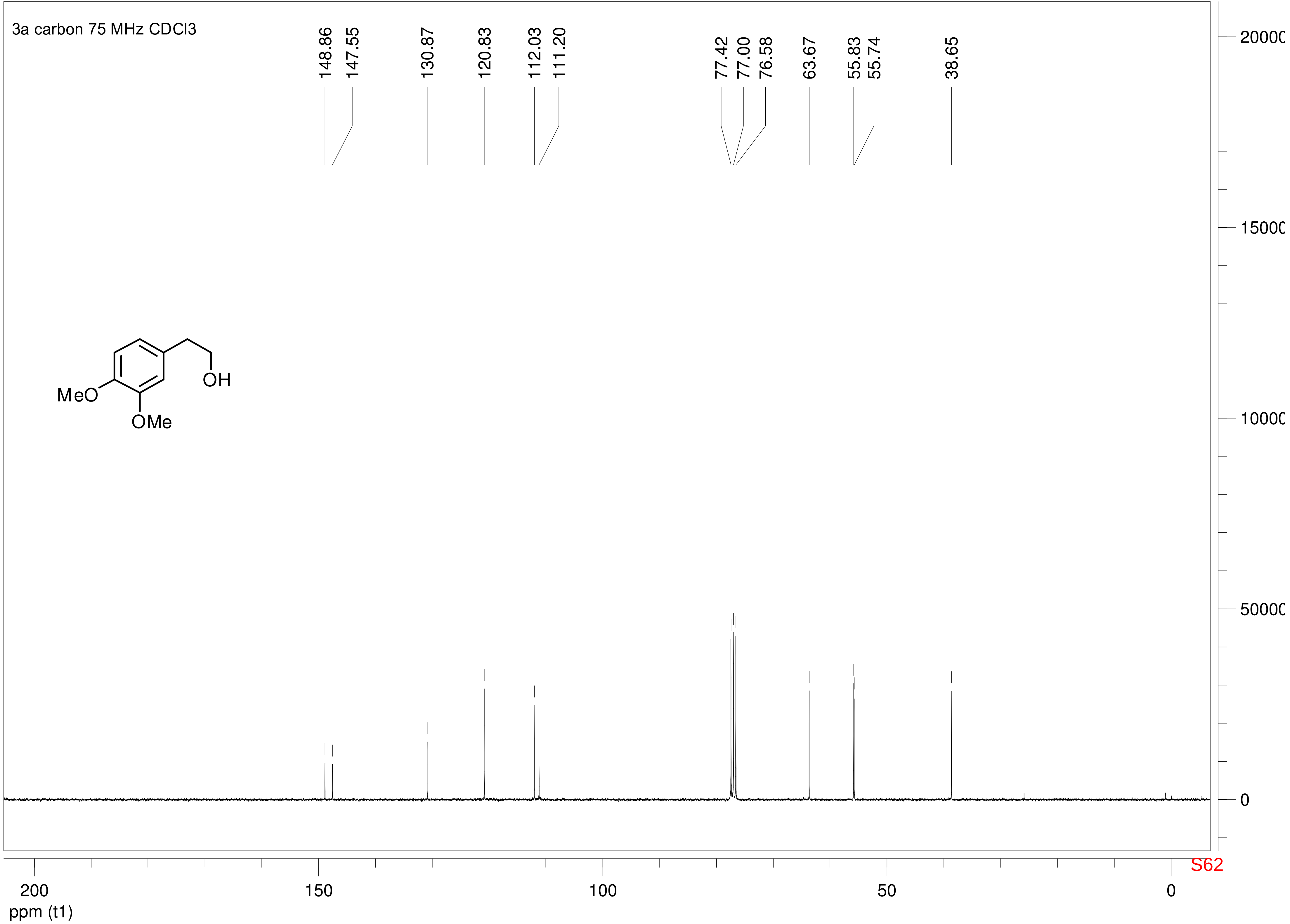




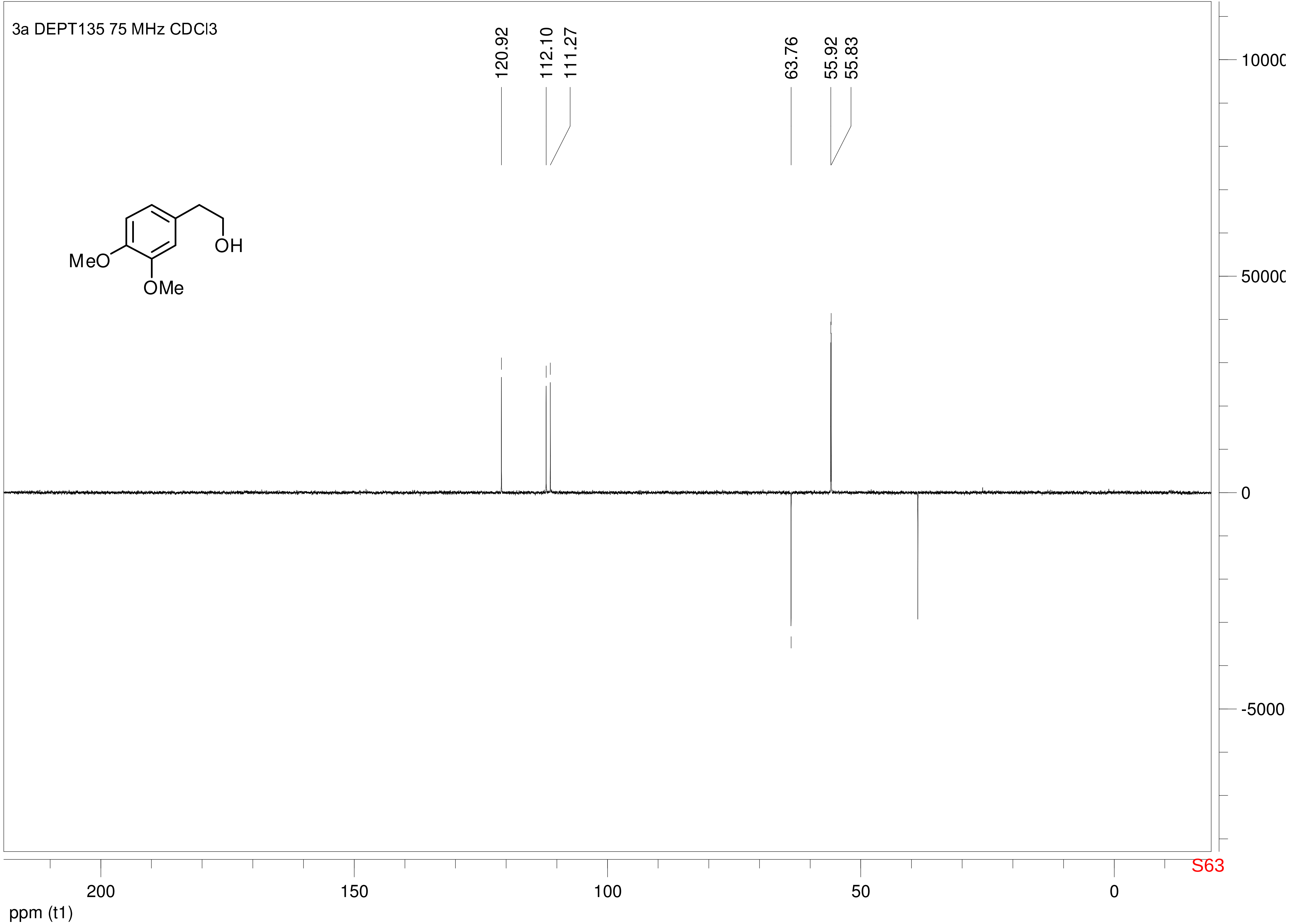

1j DEPT135 75 MHz CDCl3
128.42
128.25
125.59
63.29
35.94
32.81
31.55
29.13
26.01
25.70
-5.23
OTBS
50000
0
200
150
100
50
0
ppm (t1)

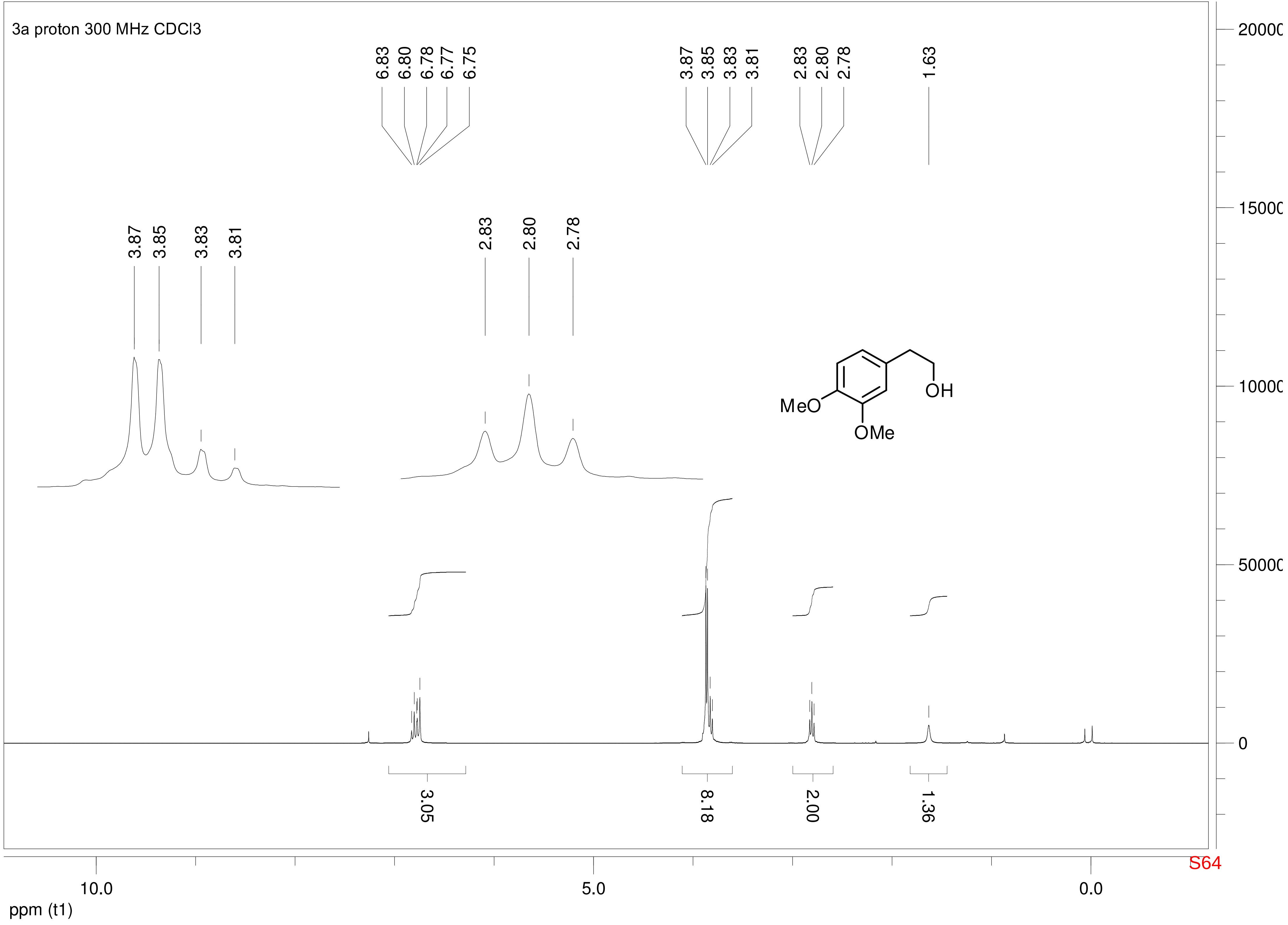
1j proton 300 MHz CDCl3



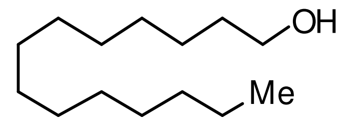


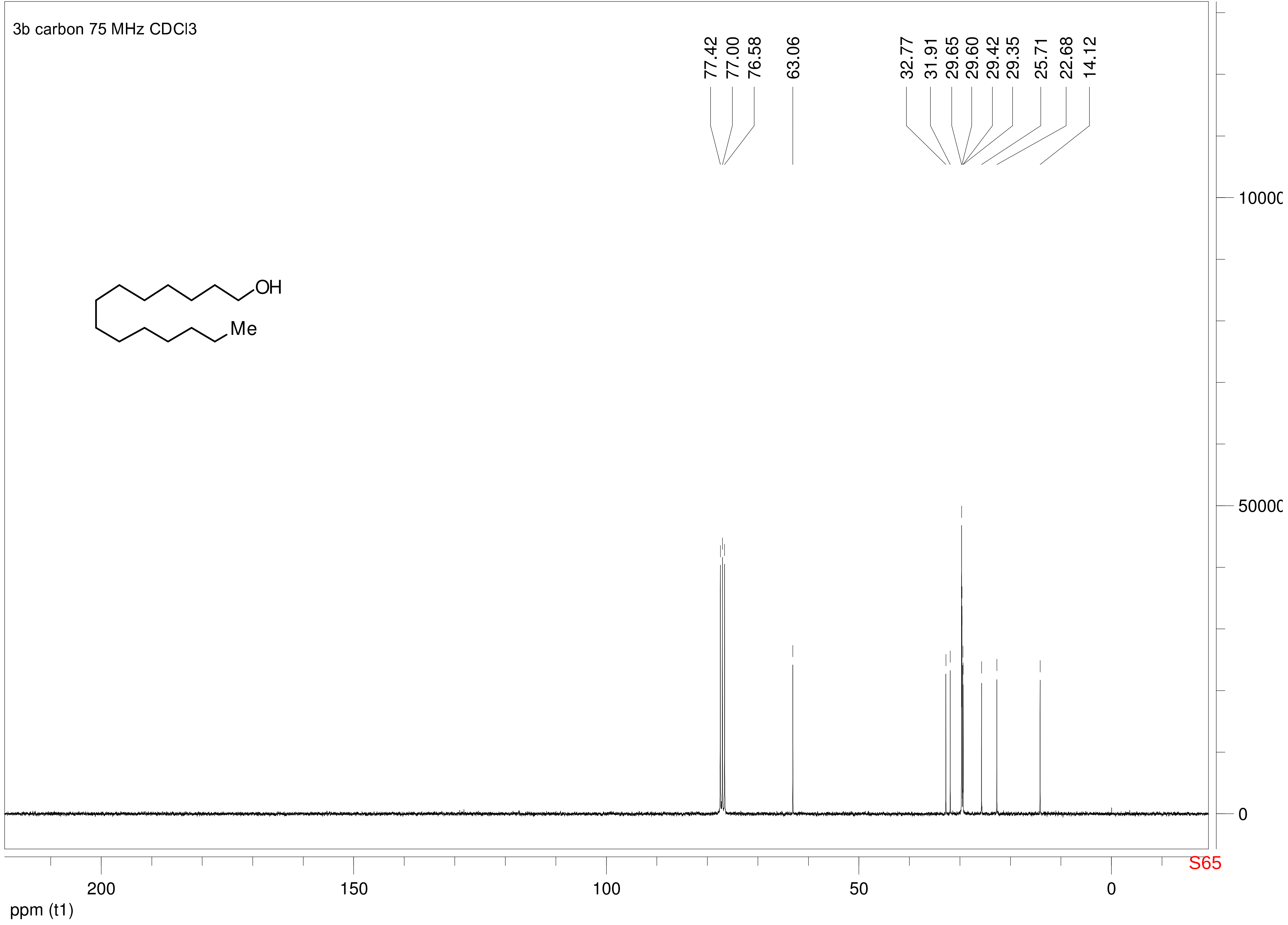
3a carbon 75 MHz CDCl3
148.86
147.55
130.87
120.83
112.03
111.20
77.42
77.00
76.58
63.67
55.83
55.74
38.65
MeO
OMe
OH
200
150
100
50
0
ppm (t1)
20000
15000
10000
50000
0


3a DEPT135 75 MHz CDCl3
120.92
112.10
111.27
63.76
55.92
55.83
MeO
OMe
OH
10000
50000
0
-5000
200
150
100
50
0
ppm (t1)

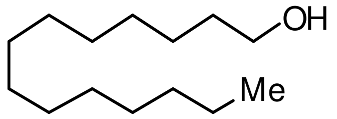

3a proton 300 MHz CDCl3
6.83
6.80
6.78
6.77
6.75
3.87
3.85
3.83
3.81
2.83
2.80
2.78
1.63
MeO
OMe
OH
3.05
8.18
2.00
1.36
10.0
5.0
0.0
ppm (t1)
20000
15000
10000
50000
0

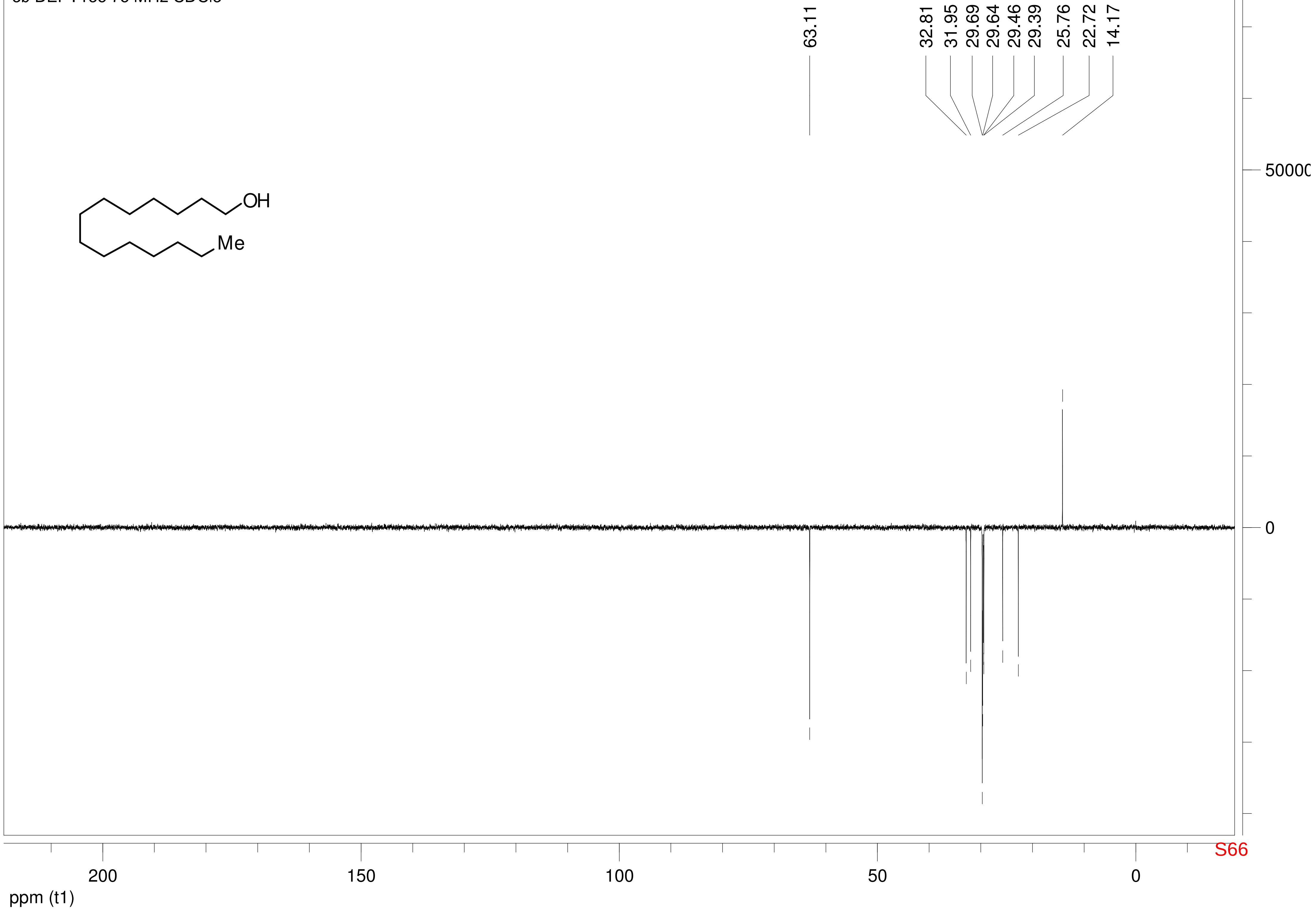
3b carbon 75 MHz CDCl3

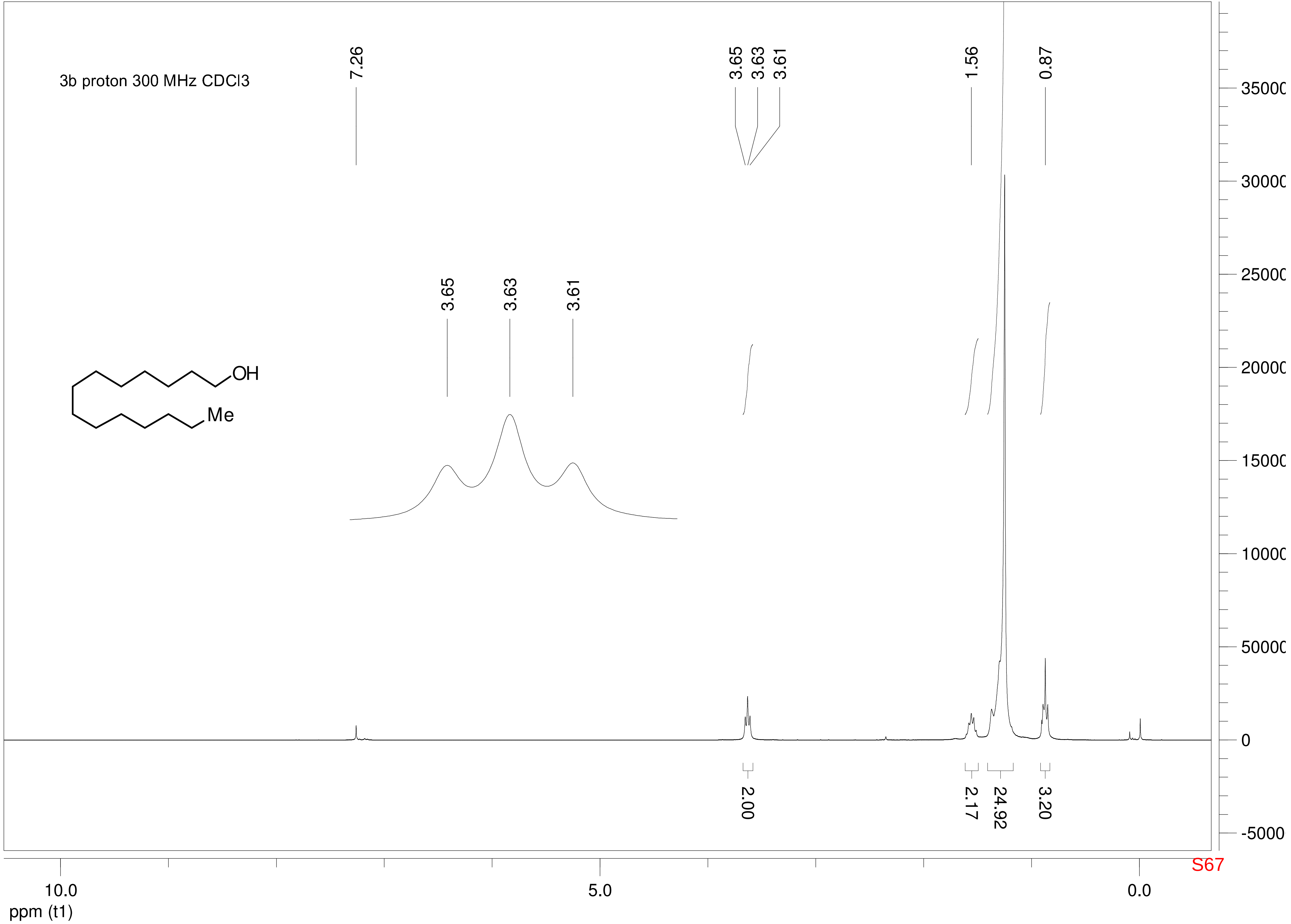




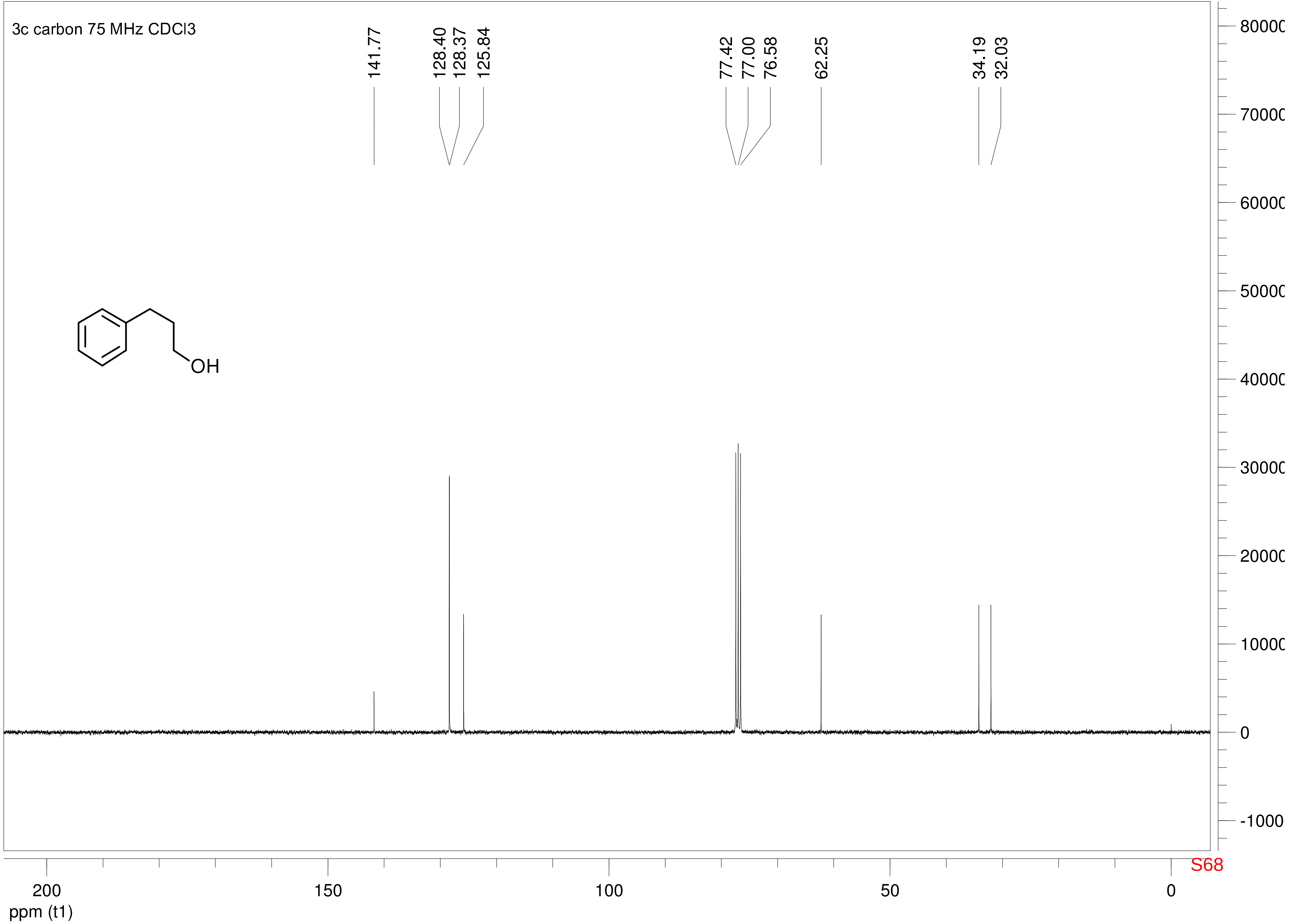
3b DEPT135 75 MHz CDCl3

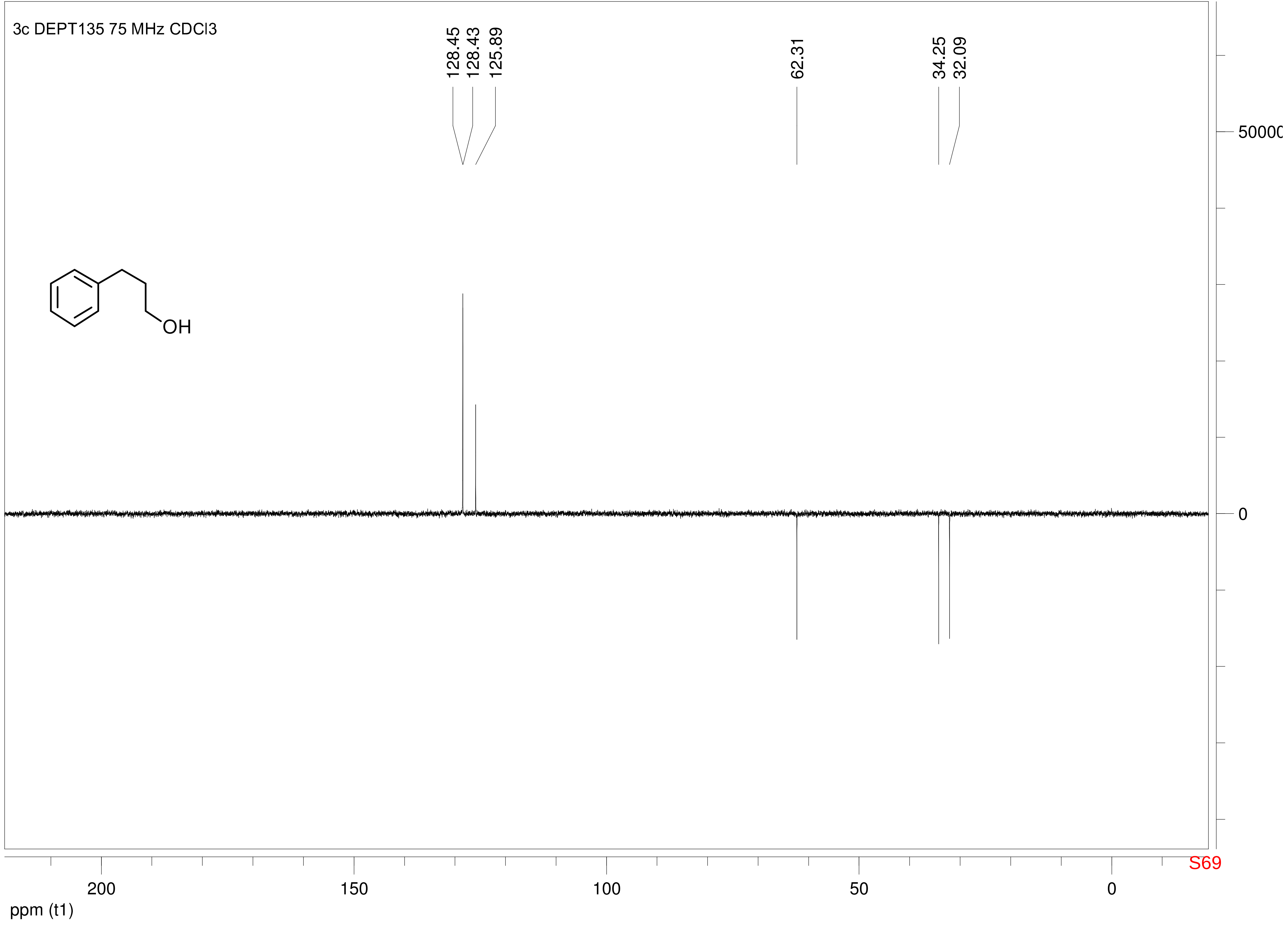




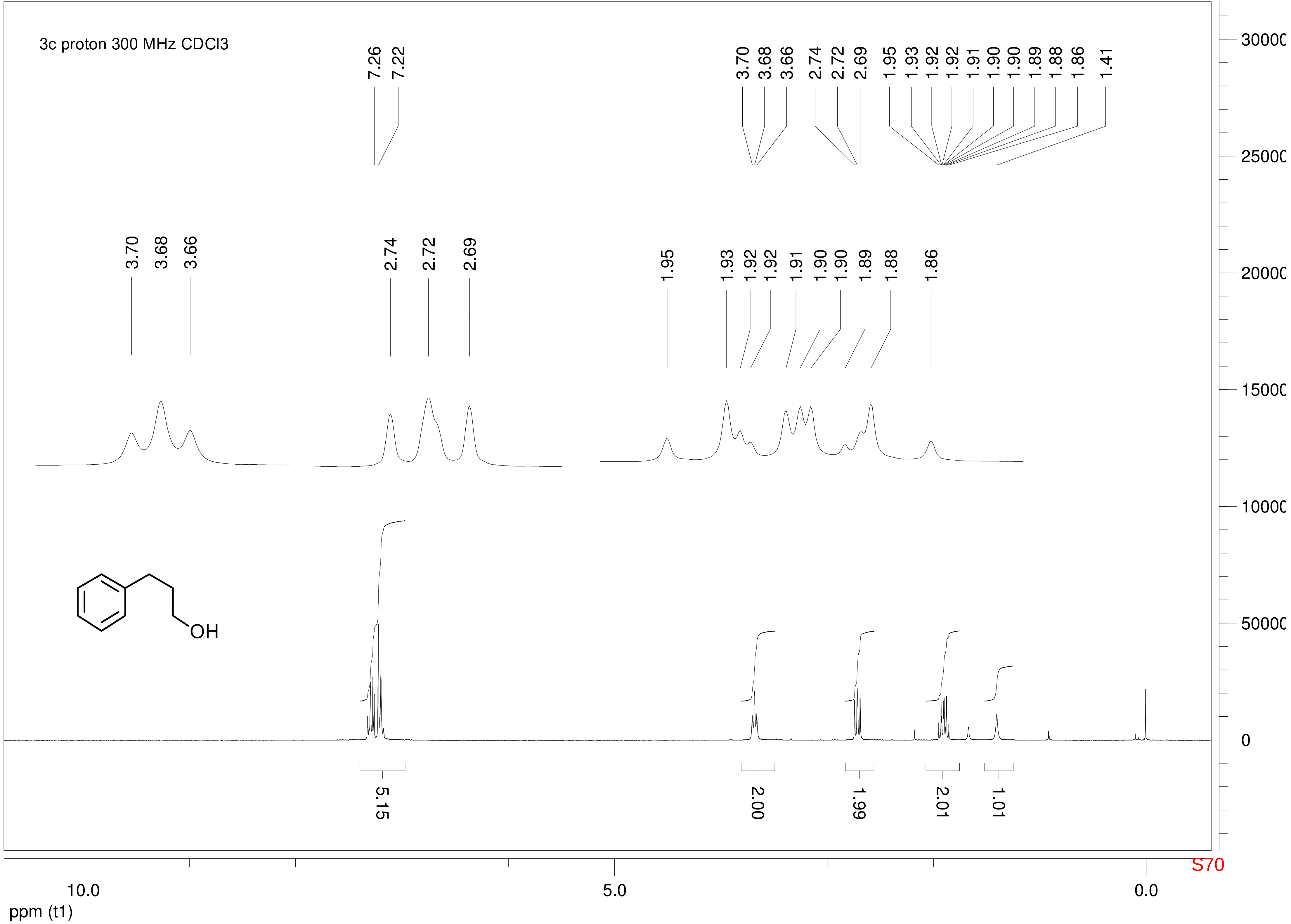


3b proton 300 MHz CDCl3
7.26
3.65
3.63
3.61
1.56
0.87
OH
Me
2.00
2.17
24.92
3.20
35000
30000
25000
20000
15000
10000
50000
0
-5000
10.0
5.0
0.0
ppm (t1)

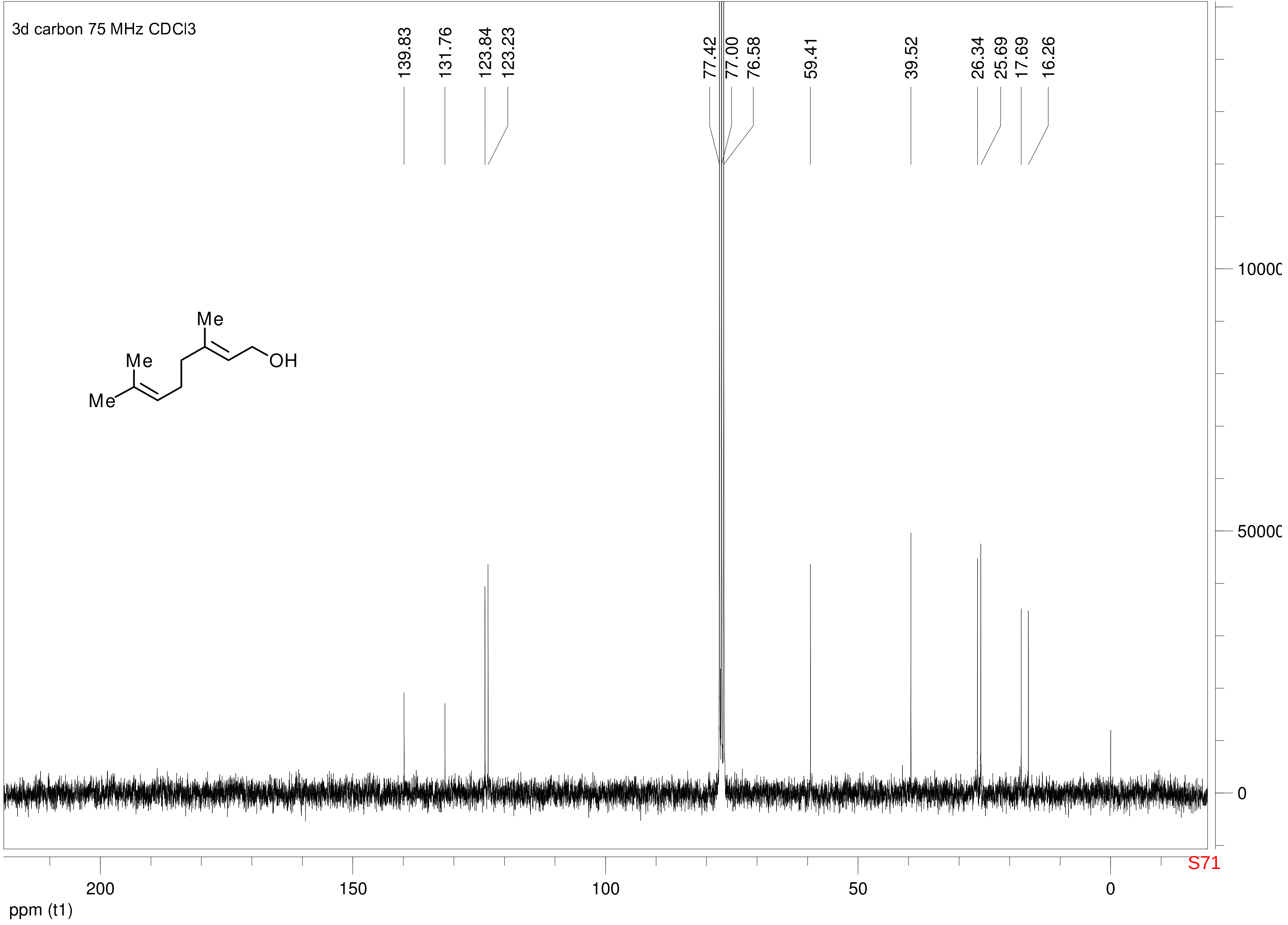

3c carbon 75 MHz CDCl3
141.77
128.40
128.37
125.84
77.42
77.00
76.58
62.25
34.19
32.03
OH
80000
70000
60000
50000
40000
30000
20000
10000
0
-1000
200
150
100
50
0
ppm (t1)



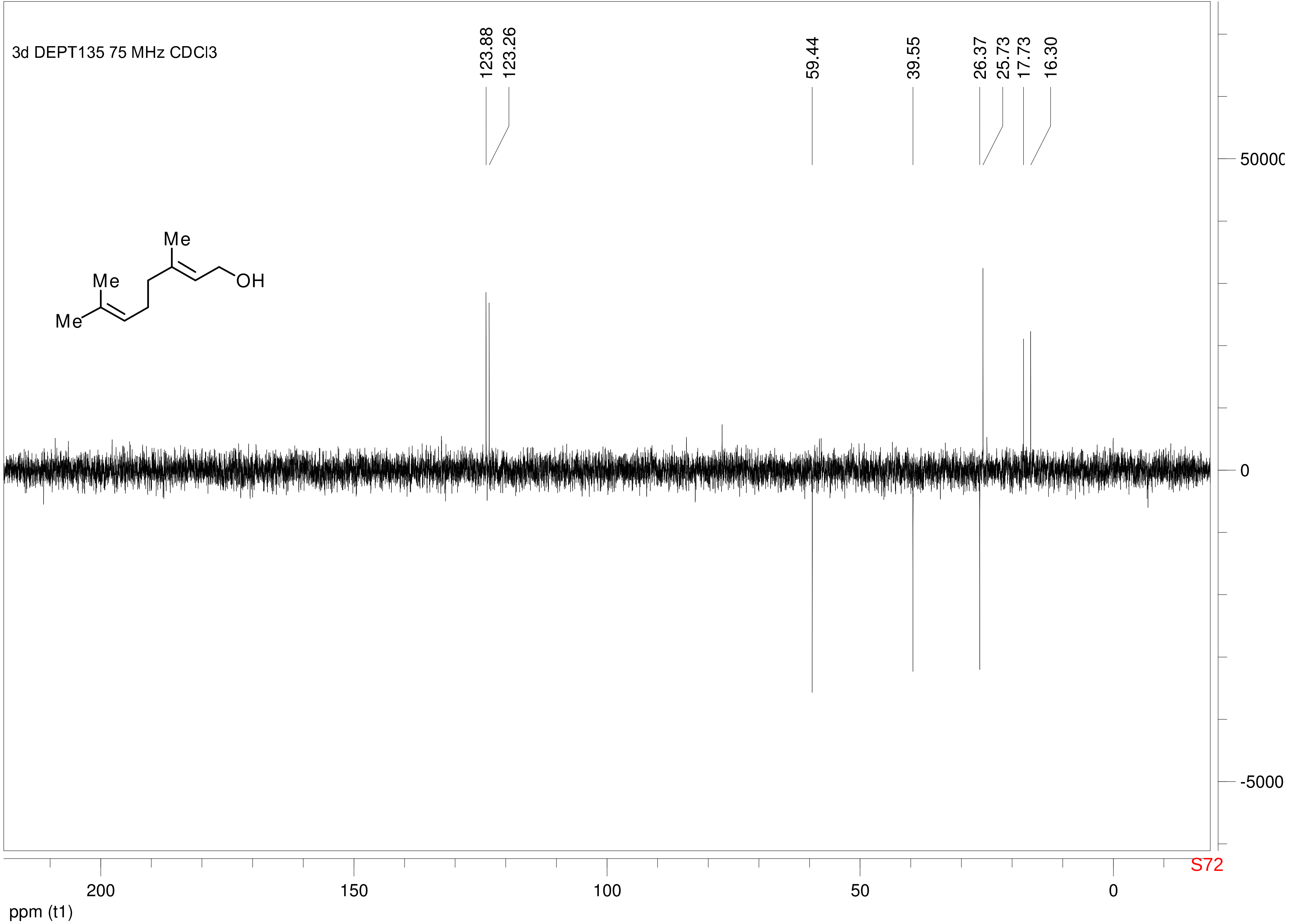
3c DEPT135 75 MHz CDCl3
128.45
128.43
125.89
62.31
34.25
32.09
OH
50000
0
200
150
100
50
0
ppm (t1)



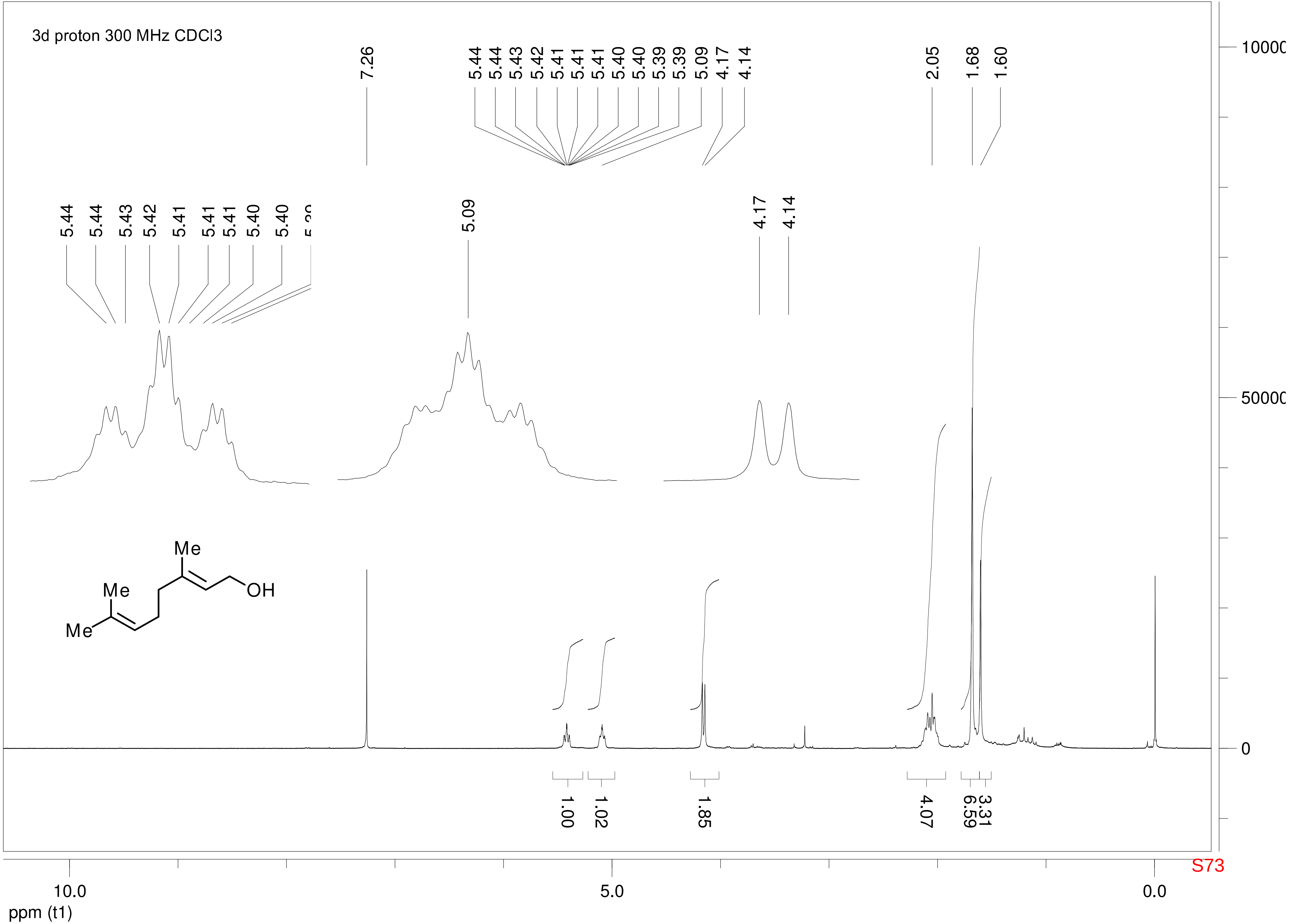
3c proton 300 MHz CDCl3
7.26
7.22
3.70
3.68
3.66
2.74
2.72
2.69
1.95
1.93
1.92
1.92
1.91
1.90
1.90
1.89
1.88
1.86
1.41
OH
5.15
2.00
1.99
2.01
1.01
10.0
5.0
0.0
ppm (t1)
30000
25000
20000
15000
10000
50000
0

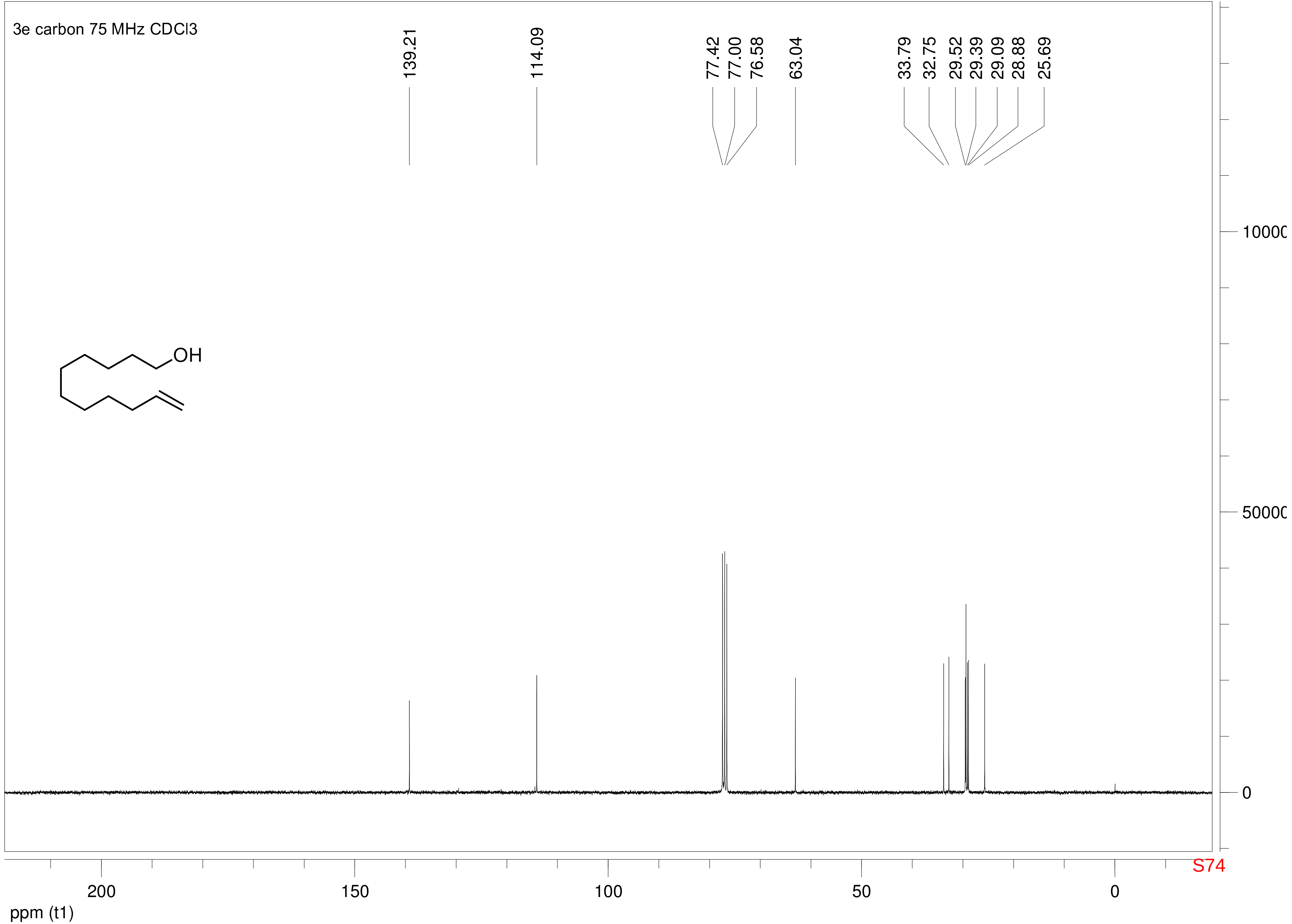


3d carbon 75 MHz CDCl3
139.83
131.76
123.84
123.23
77.42
77.00
76.58
59.41
39.52
26.34
25.69
17.69
16.26
Me
Me
Me
OH
10000
50000
0
200
150
100
50
0
ppm (t1)

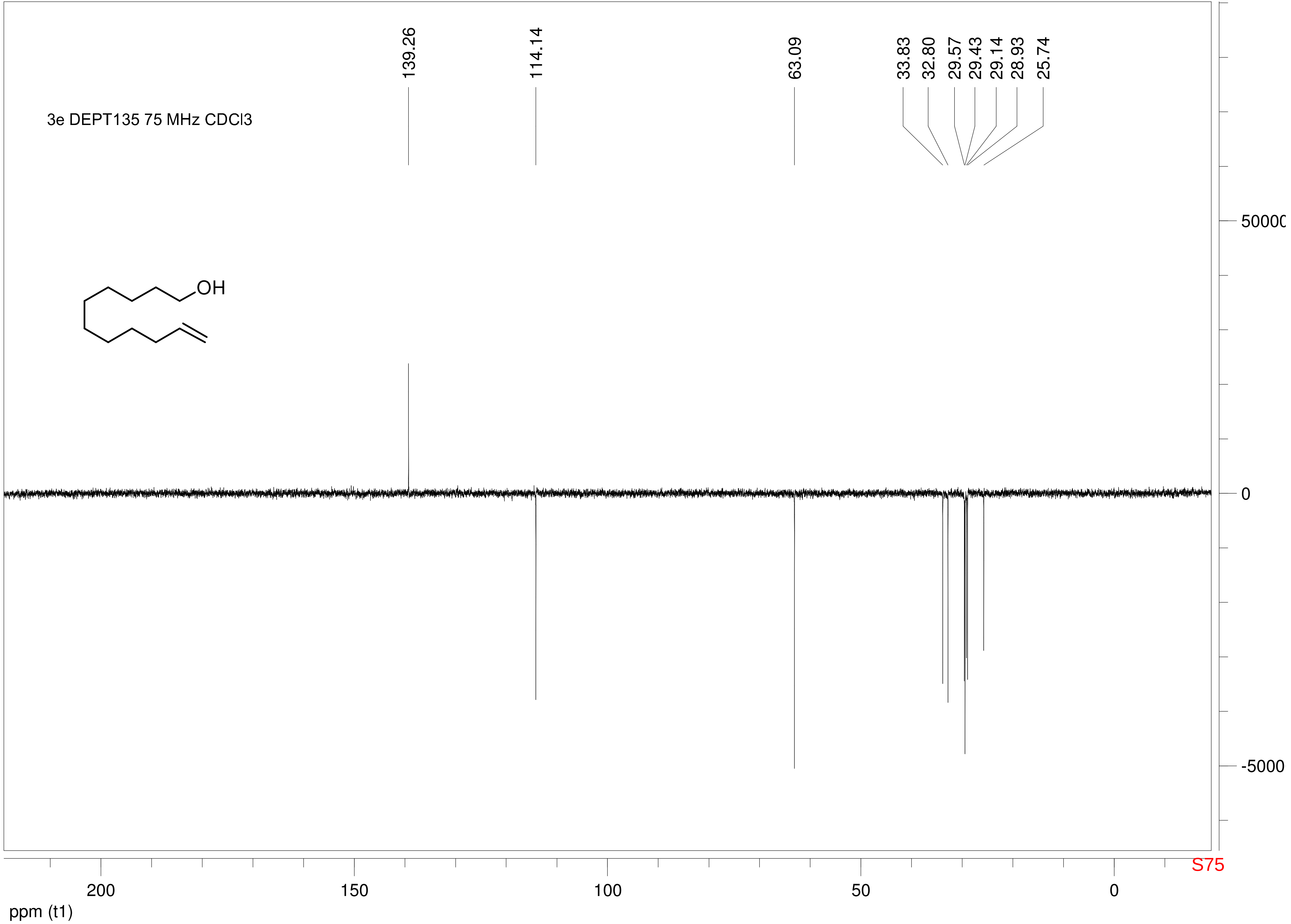


3d DEPT135 75 MHz CDCl3
123.88
123.26
59.44
39.55
26.37
25.73
17.73
16.30
Me
Me
Me
OH
50000
0
-5000
200
150
100
50
0
ppm (t1)

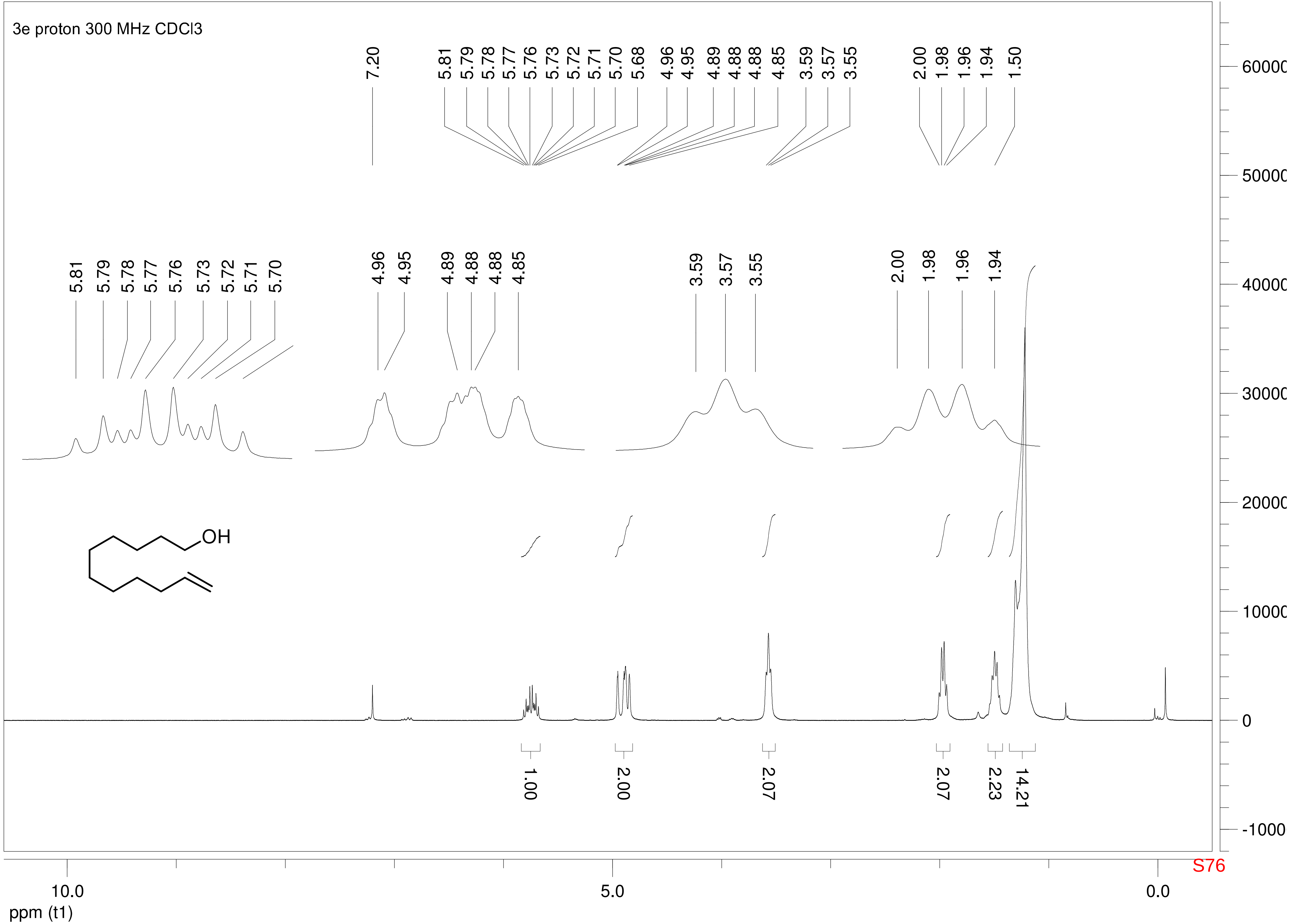

3d proton 300 MHz CDCl3
7.26
5.44 5.44 5.43 5.42 5.41 5.41 5.41 5.40 5.40 5.39 5.39 5.09 4.17 4.14
2.05 1.68 1.60
5.09
4.17 4.14
Me
Me
Me
OH
1.00 1.02 1.85 4.07 6.59 3.31
10.0
5.0
0.0
ppm (t1)
10000
50000
0



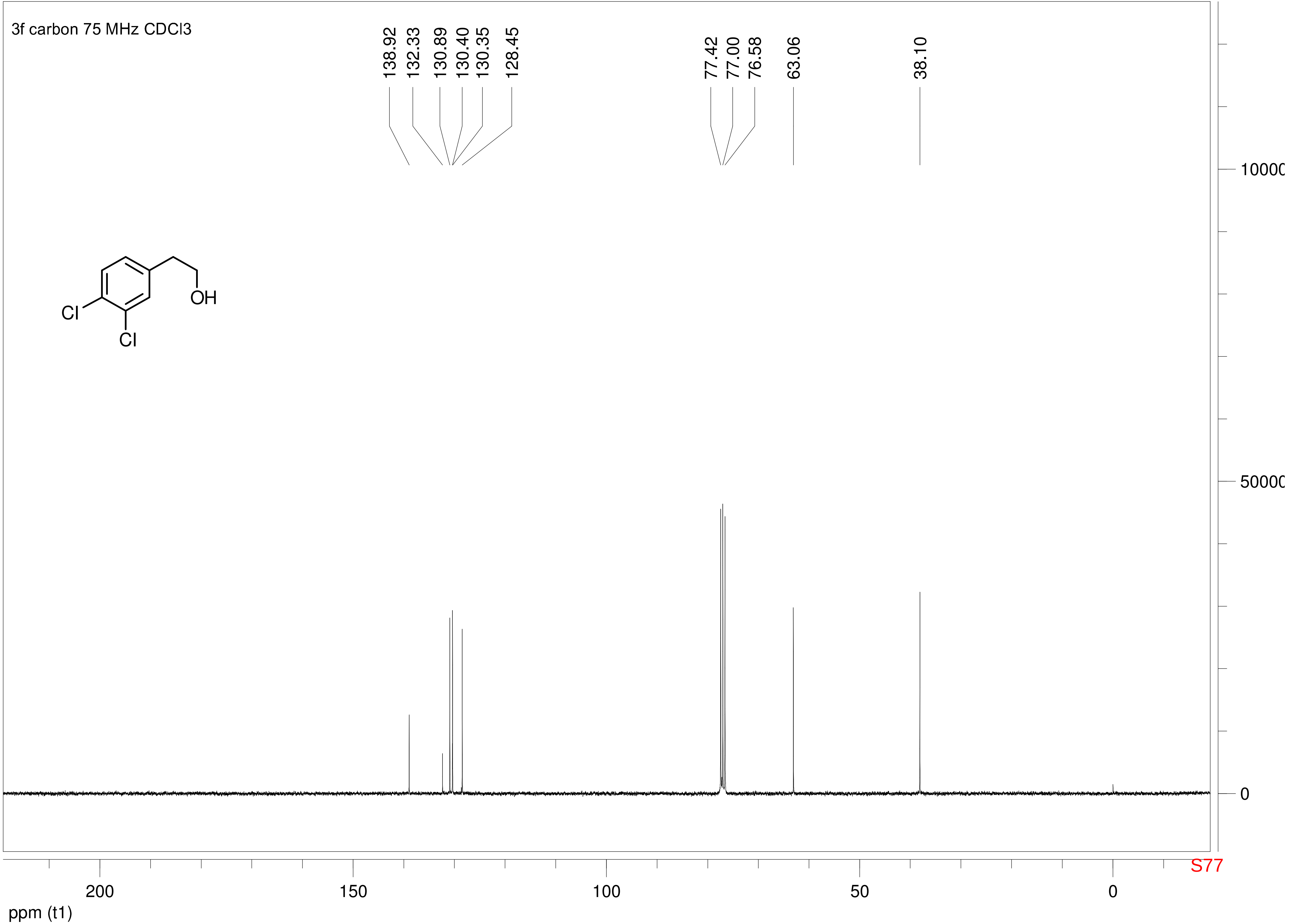
3e carbon 75 MHz CDCl3
139.21
114.09
77.42
77.00
76.58
63.04
33.79
32.75
29.52
29.39
29.09
28.88
25.69
OH
10000
50000
0
200
150
100
50
0
ppm (t1)



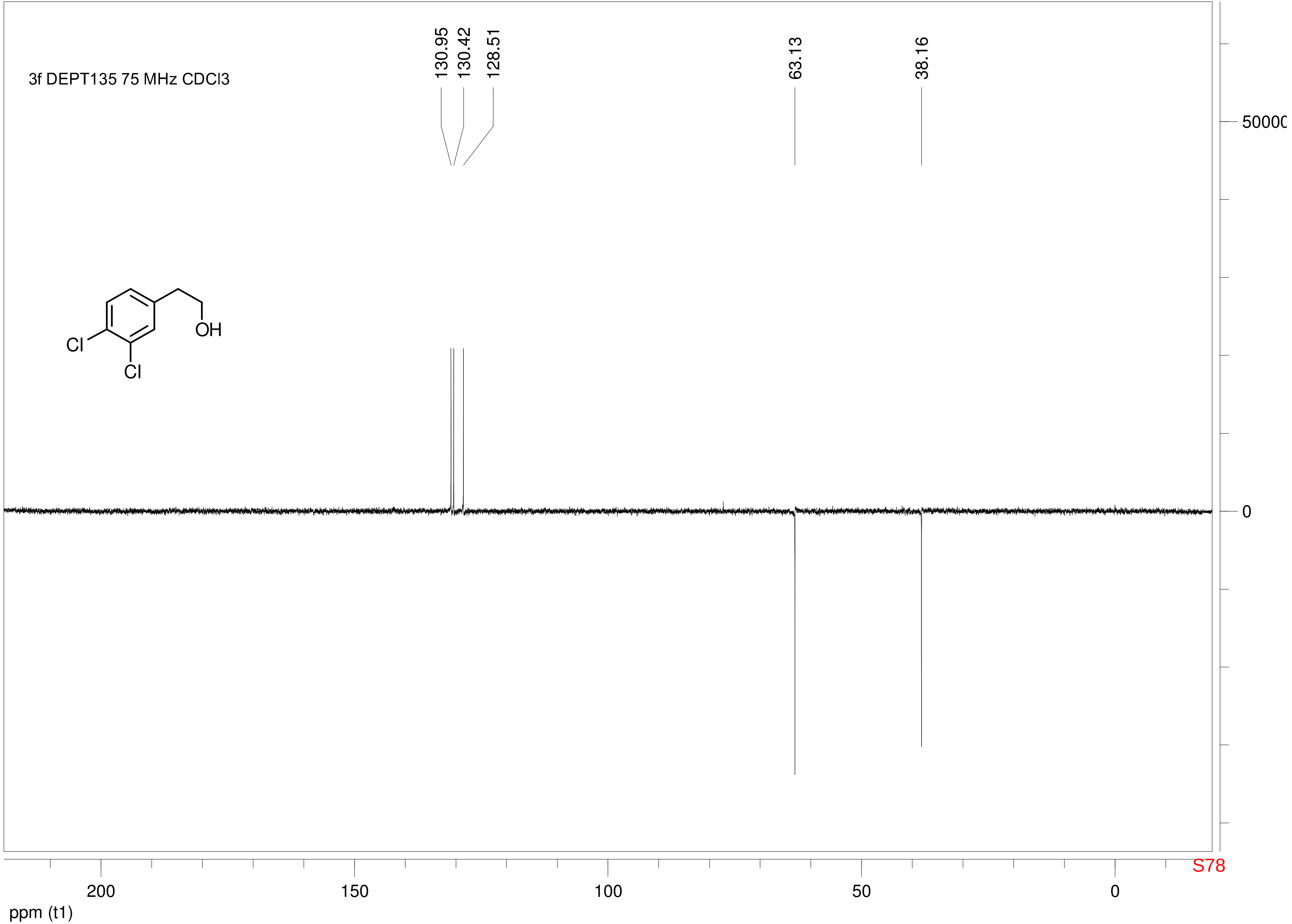
139.26
114.14
63.09
33.83
32.80
29.57
29.43
29.14
28.93
25.74
3e DEPT135 75 MHz CDCl3
OH
50000
0
-5000
200
150
100
50
0
ppm (t1)

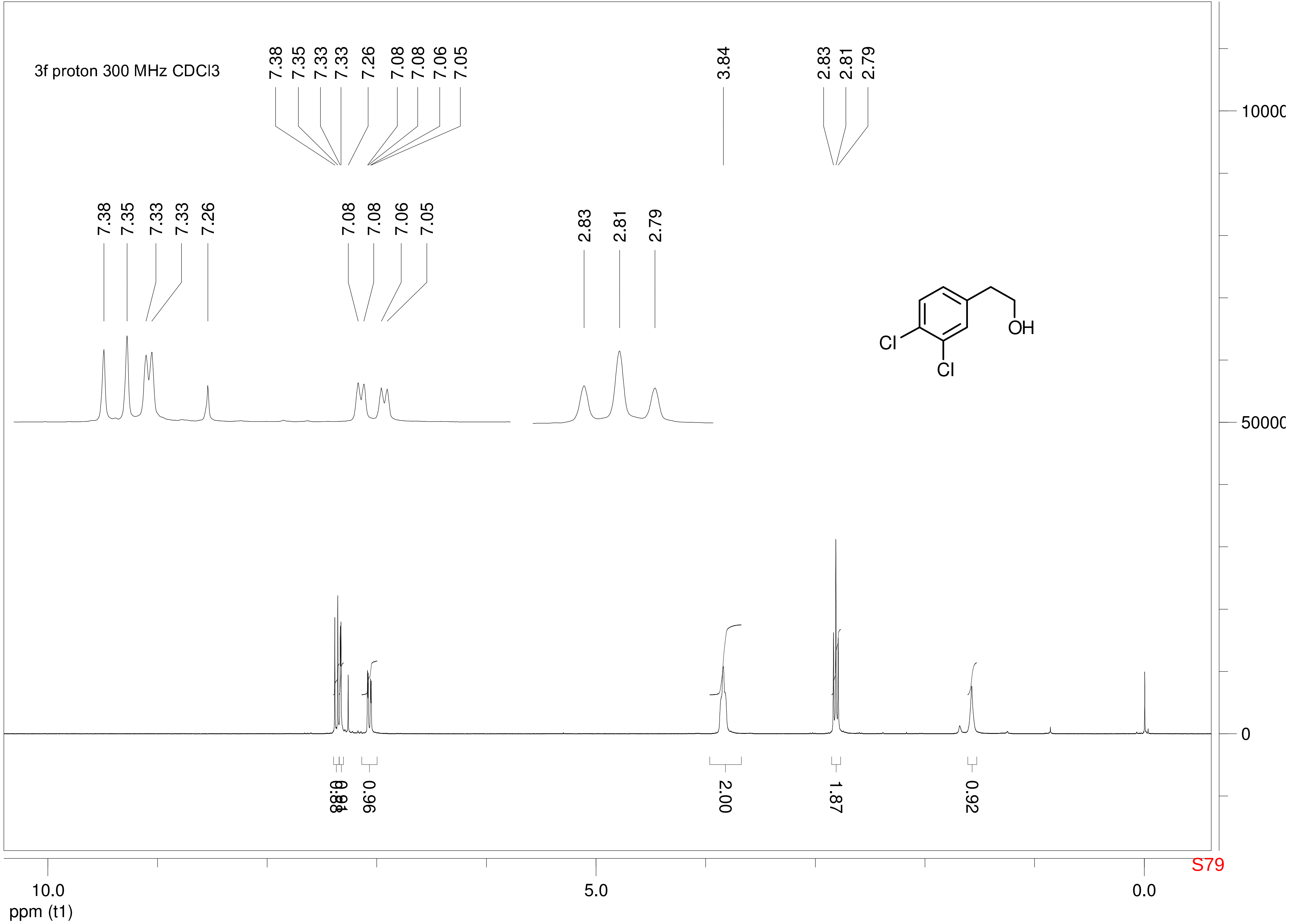


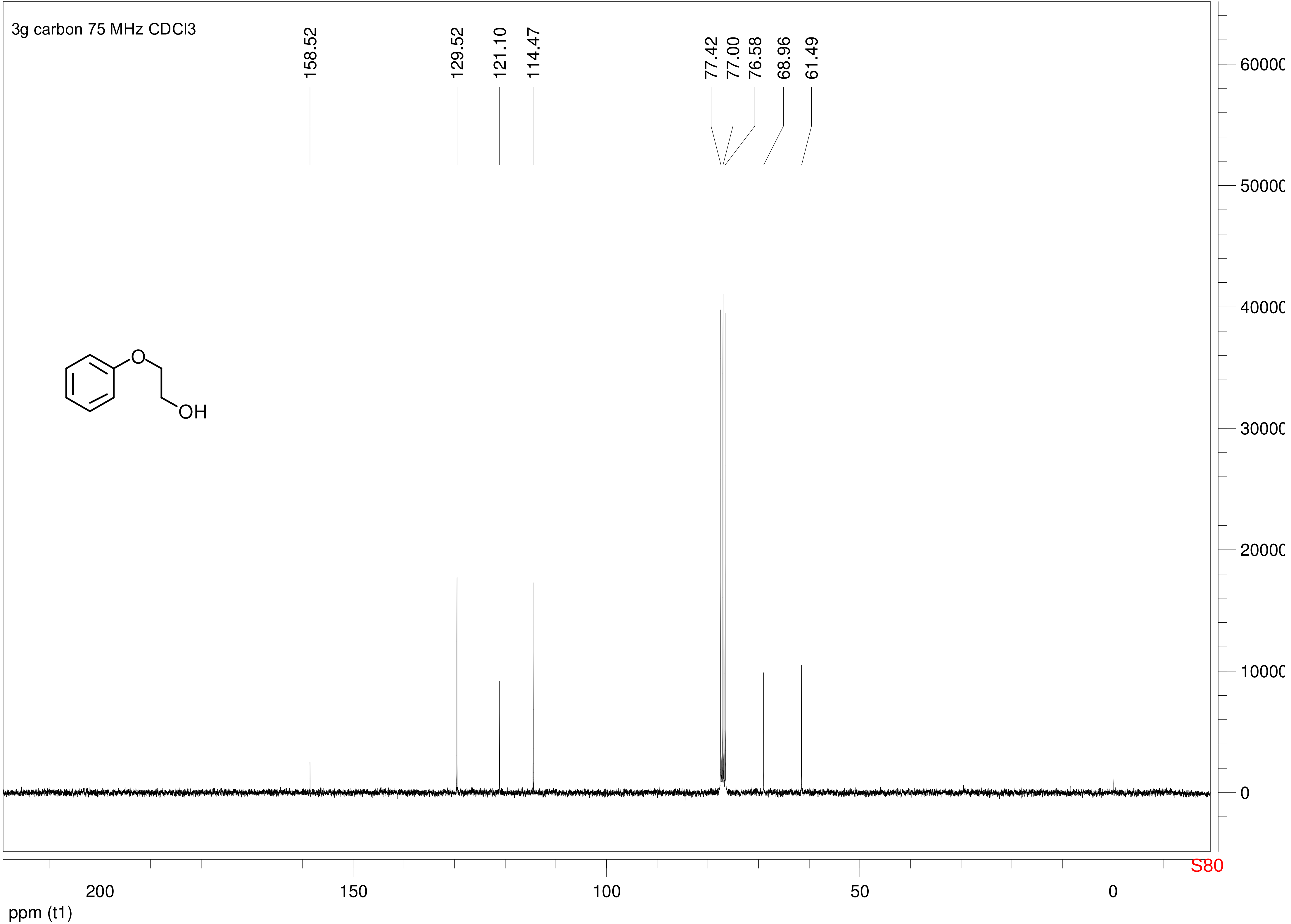
3e proton 300 MHz CDCl3
7.20
5.81
5.79
5.78
5.77
5.76
5.73
5.72
5.71
5.70
5.68
4.96
4.95
4.89
4.88
4.88
4.85
3.59
3.57
3.55
2.00
1.98
1.96
1.94
1.50
OH
1.00
2.00
2.07
2.07
2.23
14.21
10.0
5.0
0.0
ppm (t1)
60000
50000
40000
30000
20000
10000
0
-1000



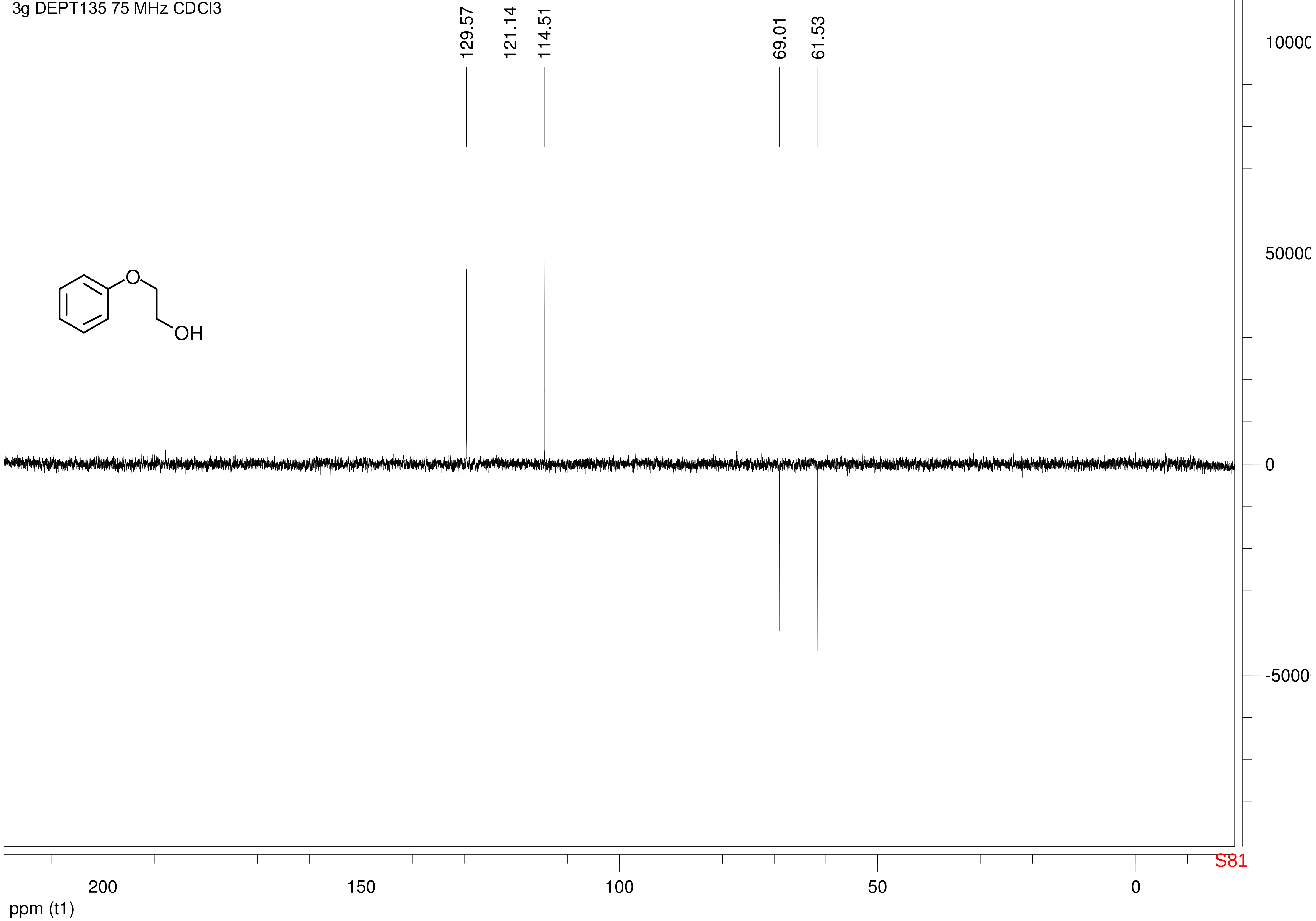
3f carbon 75 MHz CDCl3
138.92
132.33
130.89
130.40
130.35
128.45
77.42
77.00
76.58
63.06
38.10
Cl
Cl
OH
10000
50000
0
200
150
100
50
0
ppm (t1)

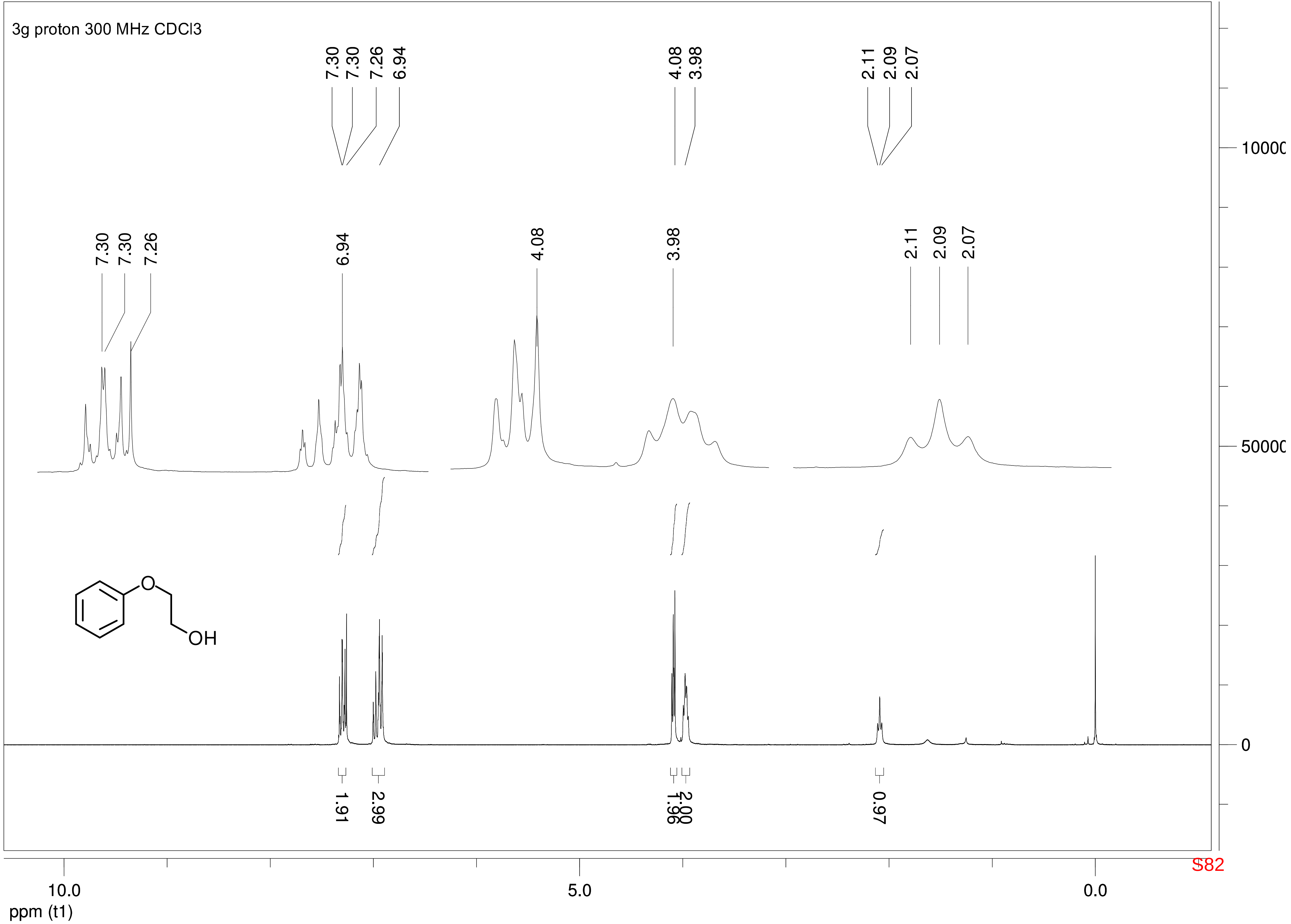

3f DEPT135 75 MHz CDCl3
130.95
130.42
128.51
63.13
38.16
50000
0
Cl
Cl
OH
200
150
100
50
0
ppm (t1)

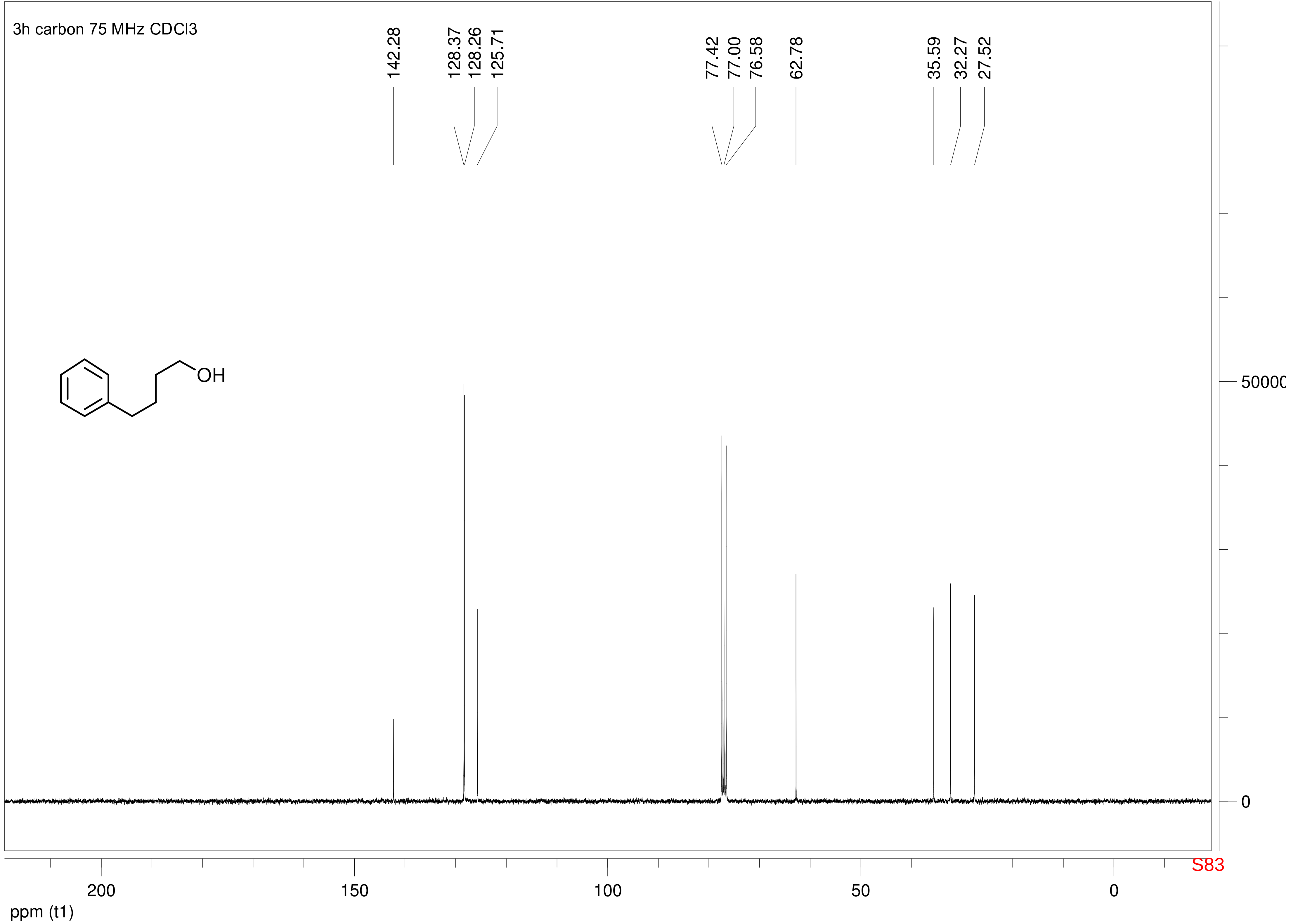

3f proton 300 MHz CDCl3
7.38
7.35
7.33
7.33
7.26
7.08
7.08
7.06
7.05
3.84
2.83
2.81
2.79
Cl
Cl
OH
10000
50000
0
0.88
0.91
0.84
0.96
2.00
1.87
0.92
10.0
5.0
0.0
ppm (t1)

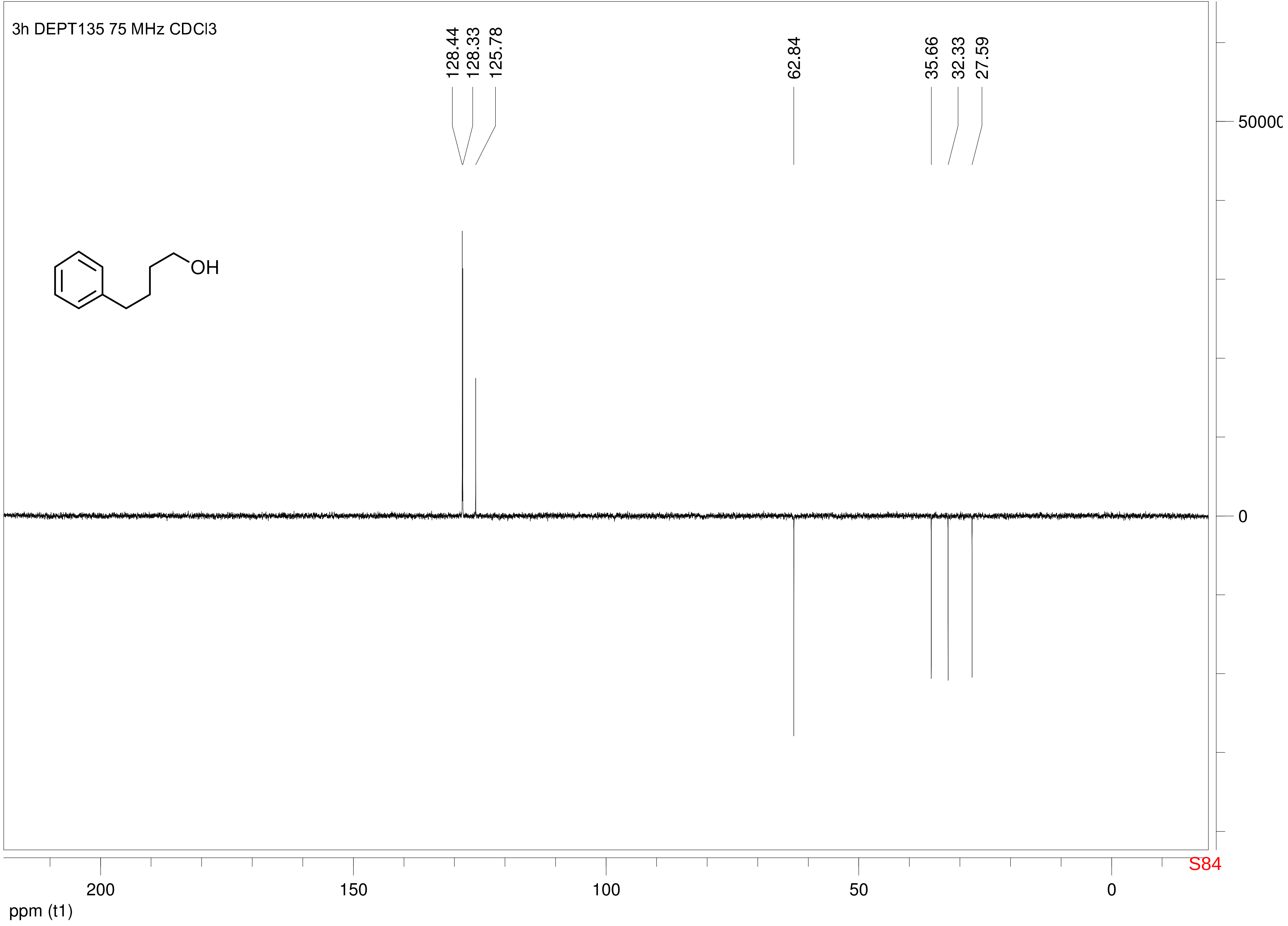

3g carbon 75 MHz CDCl3
158.52
129.52
121.10
114.47
77.42
77.00
76.58
68.96
61.49
OH
O
60000
50000
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)

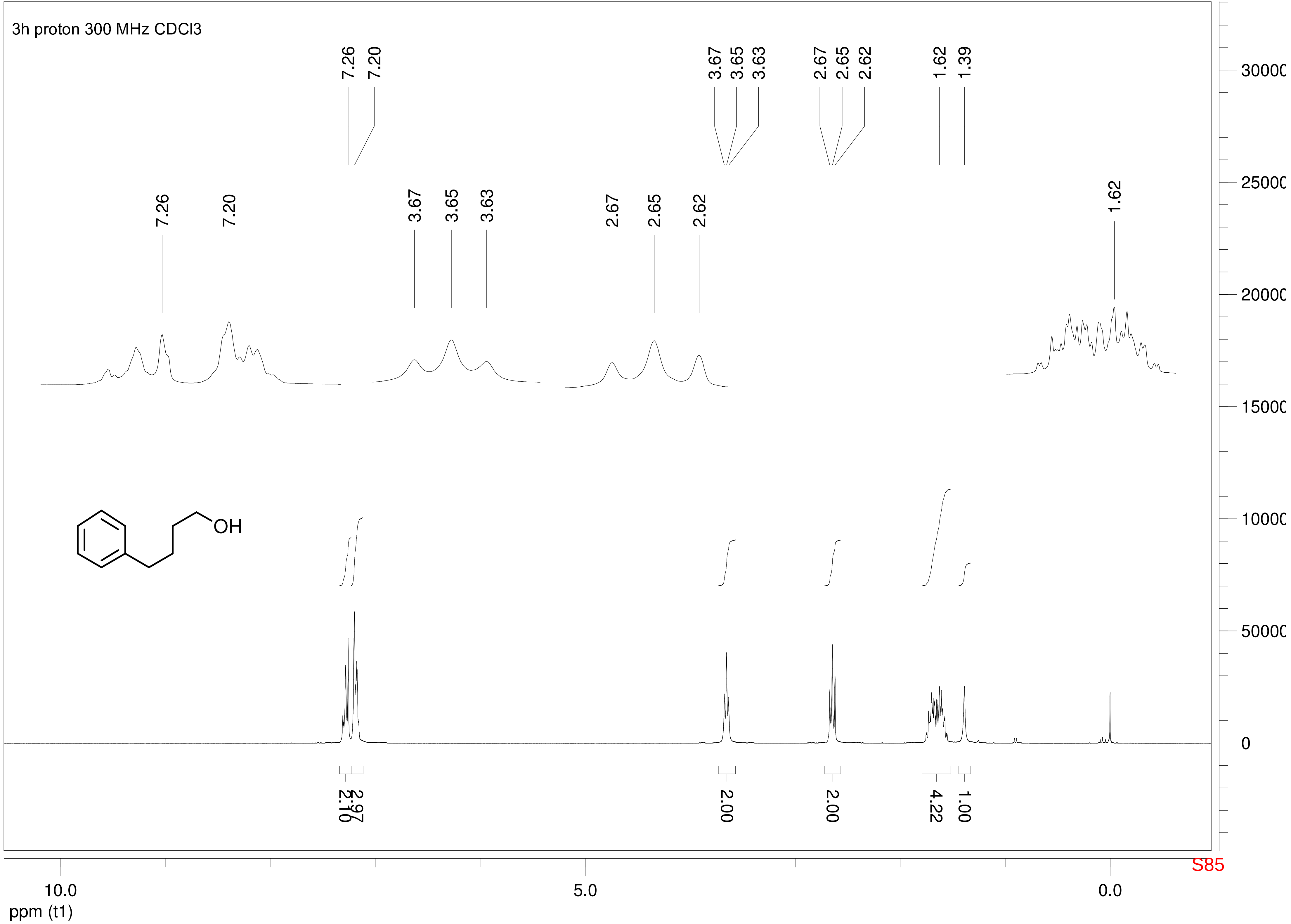
3g DEPT135 75 MHz CDCl3

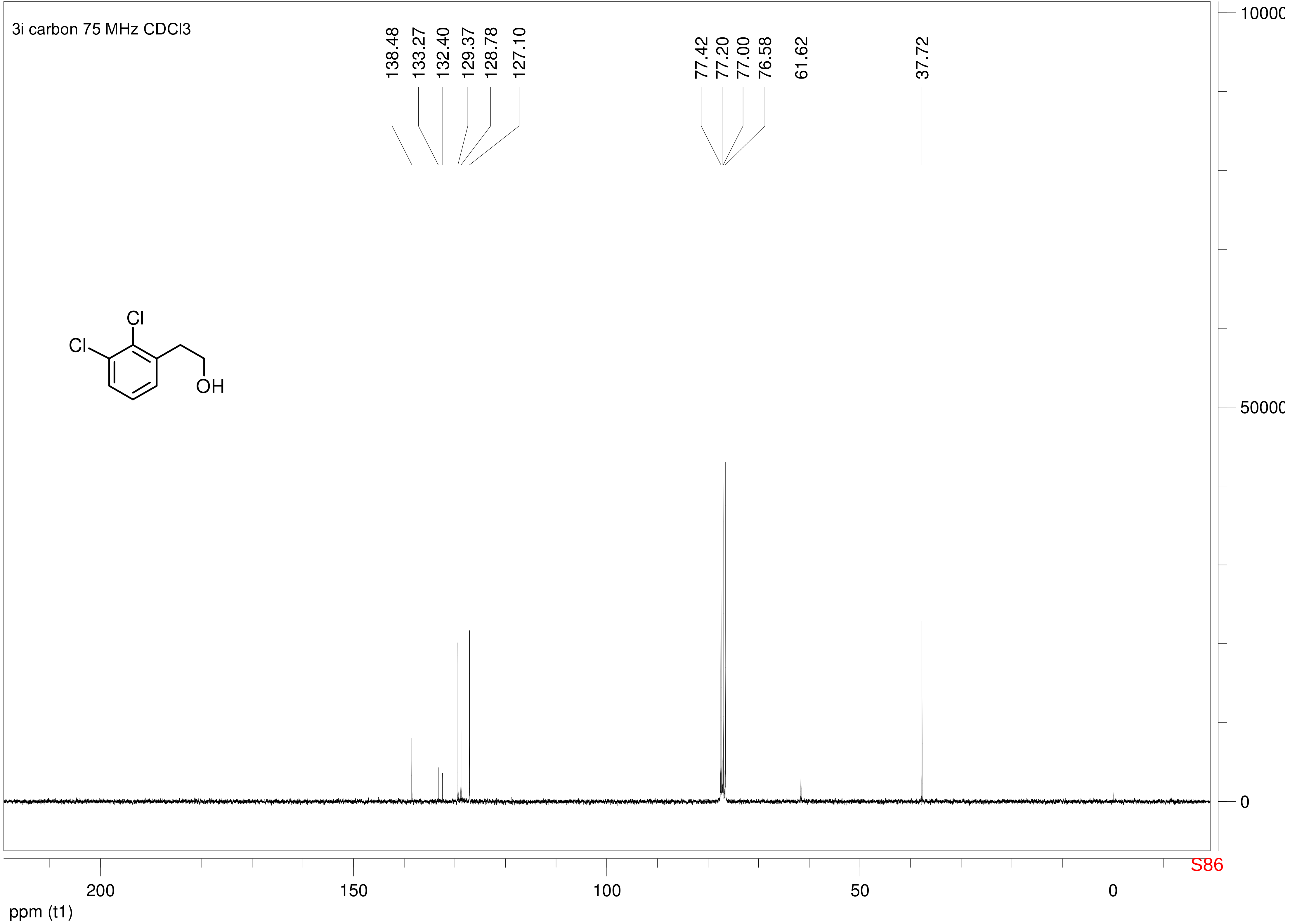


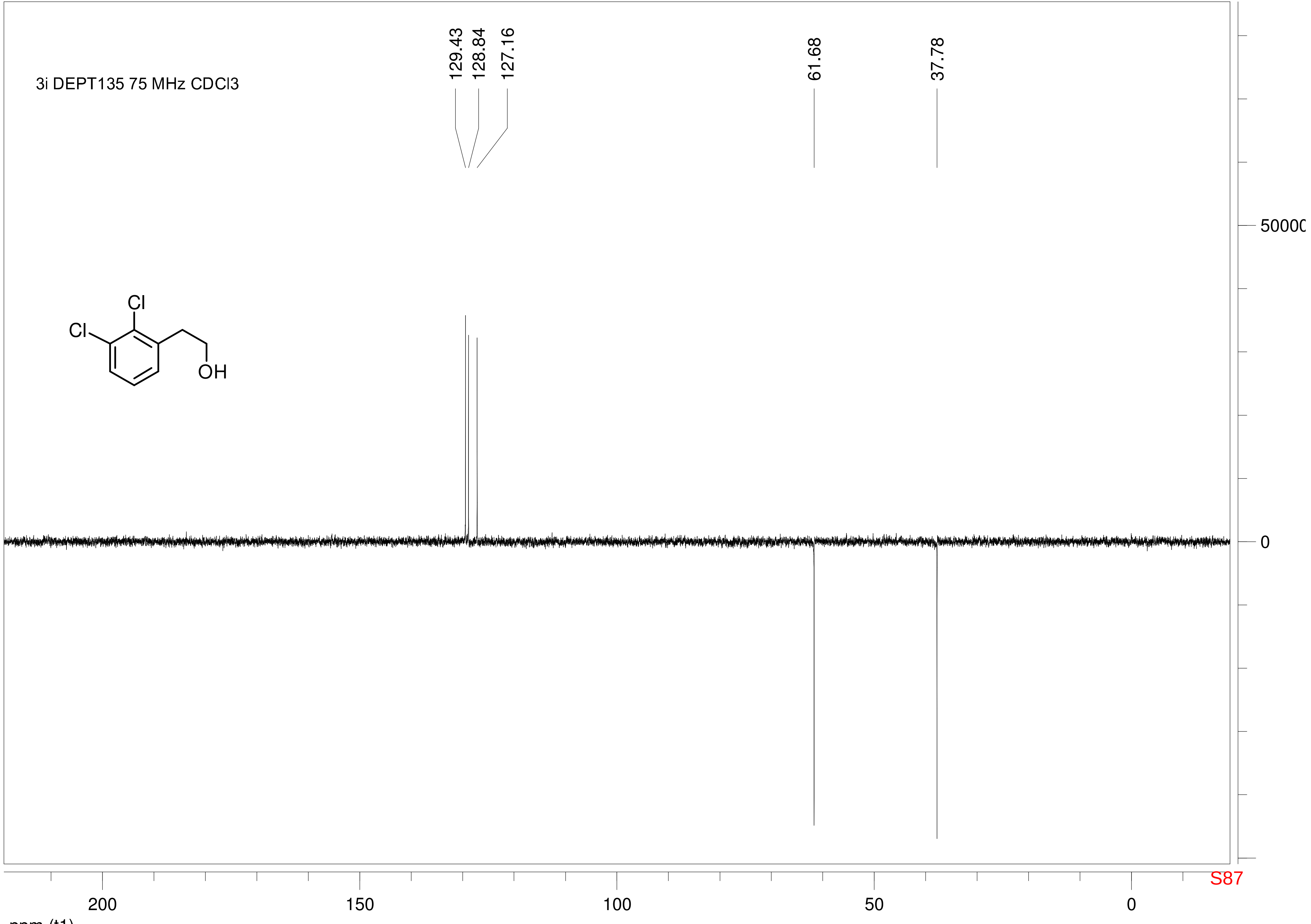

3g proton 300 MHz CDCl3
7.30
7.30
7.26
6.94
4.08
3.98
2.11
2.09
2.07
OH
O
1.91
2.99
1.96
2.00
0.97
10.0
5.0
0.0
ppm (t1)
100000
50000
0

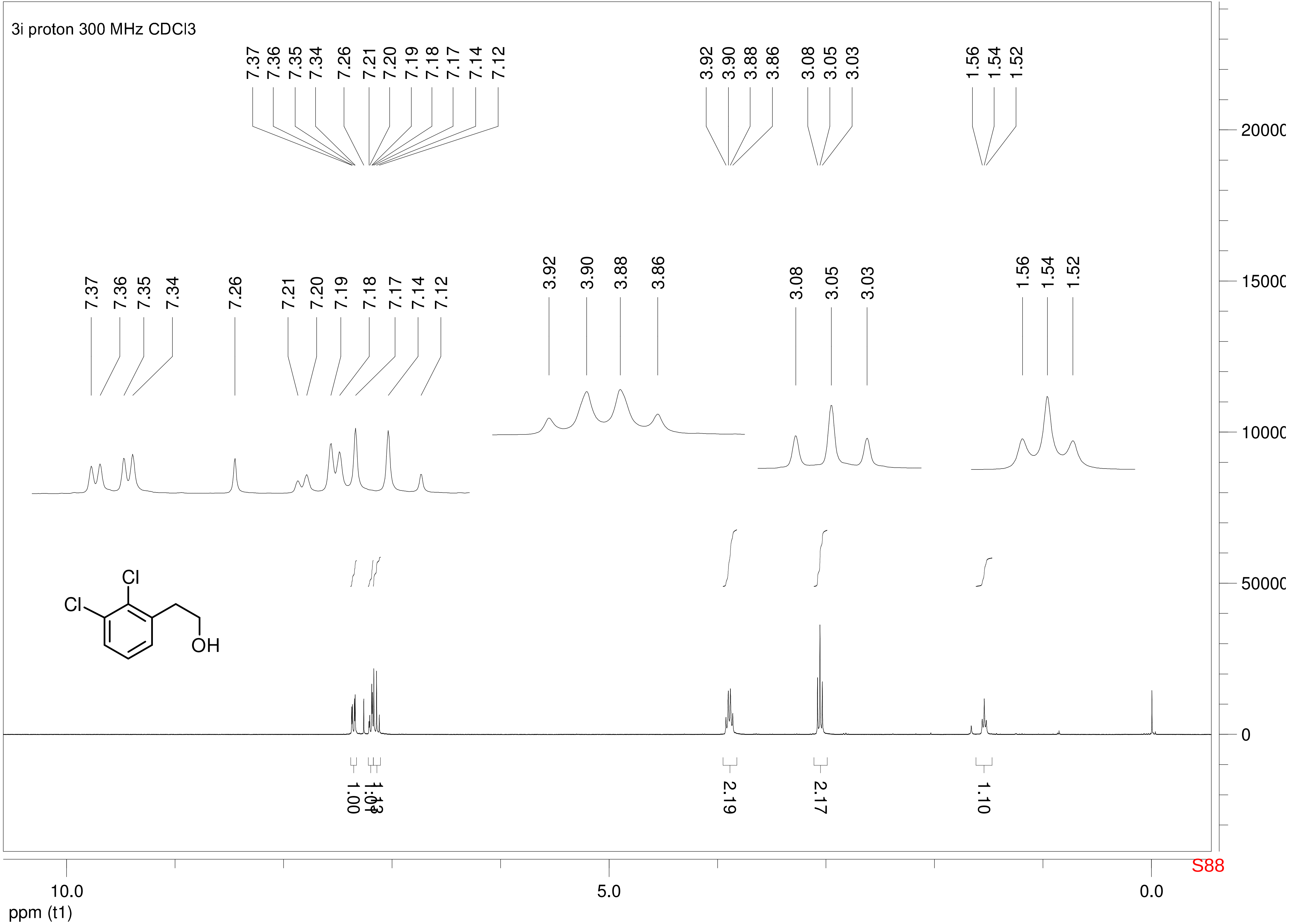

3h carbon 75 MHz CDCl3
142.28
128.37
128.26
125.71
77.42
77.00
76.58
62.78
35.59
32.27
27.52
OH
50000
0
200
150
100
50
0
ppm (t1)


3h DEPT135 75 MHz CDCl3
128.44
128.33
125.78
62.84
35.66
32.33
27.59
OH
50000
0
200
150
100
50
0
ppm (t1)

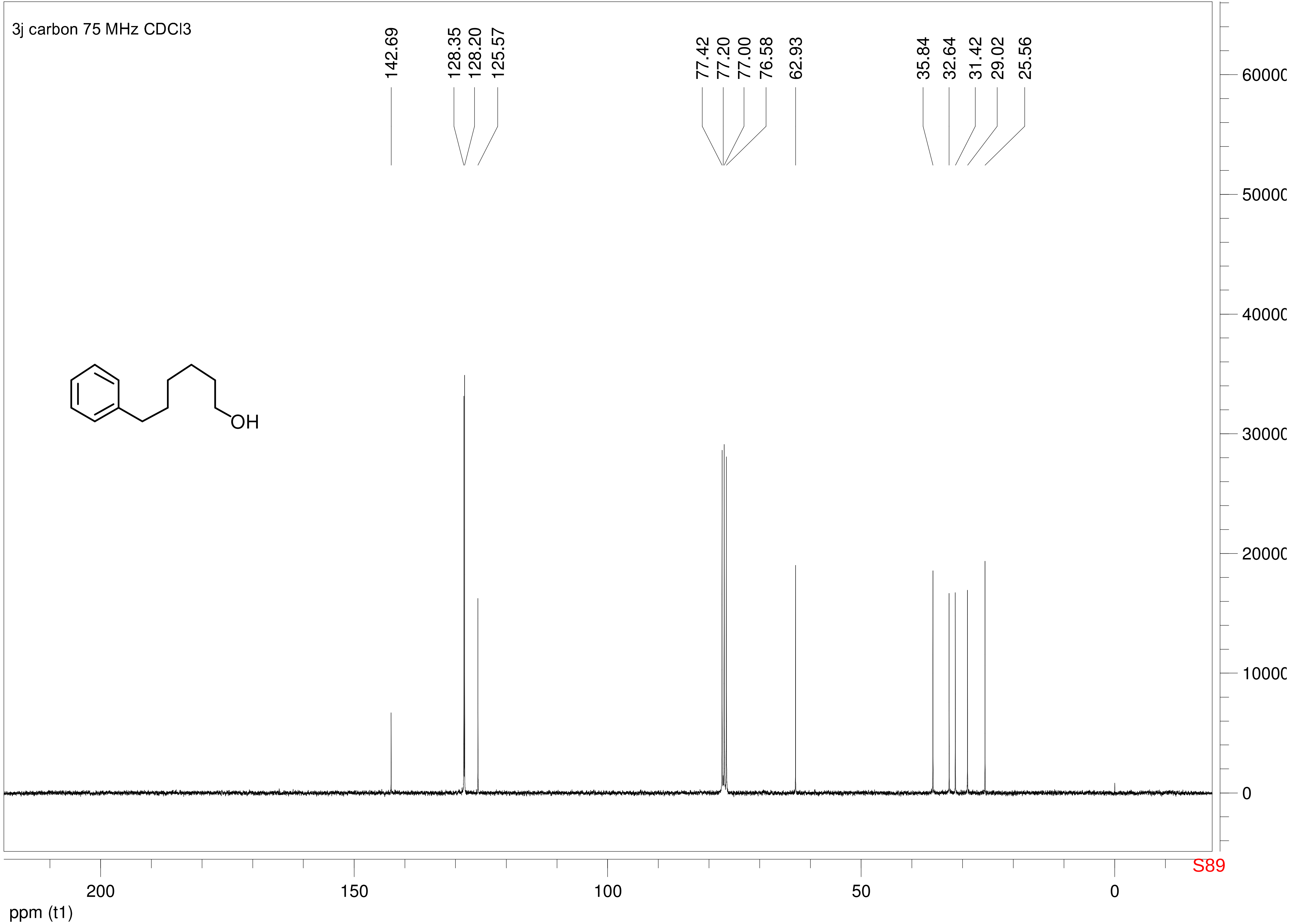

3h proton 300 MHz CDCl3
7.26
7.20
3.67
3.65
3.63
2.67
2.65
2.62
1.62
1.39
OH
2.10
2.97
2.00
2.00
4.22
1.00
10.0
5.0
0.0
ppm (t1)
30000
25000
20000
15000
10000
50000
0

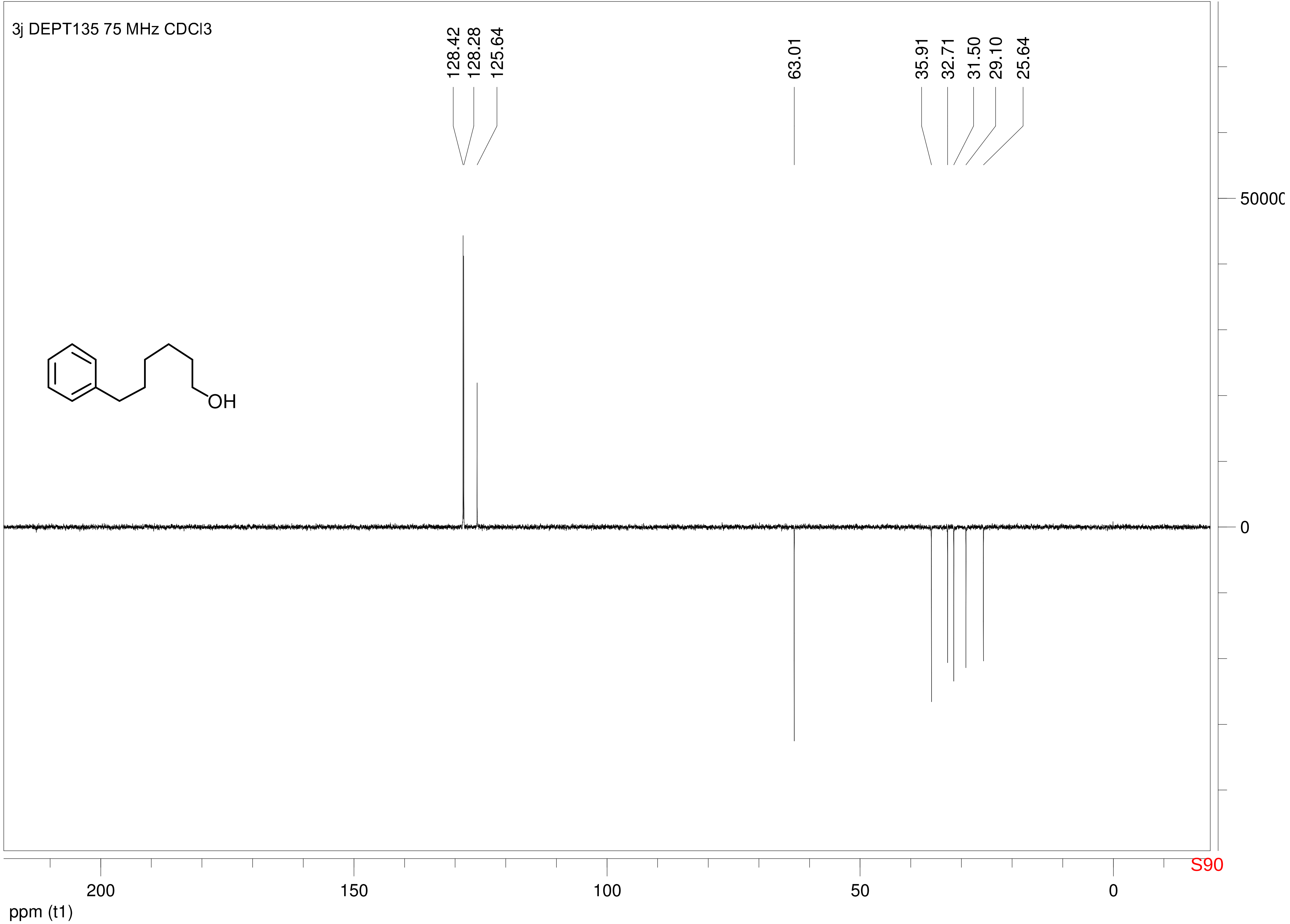

3i carbon 75 MHz CDCl3
138.48
133.27
132.40
129.37
128.78
127.10
77.42
77.20
77.00
76.58
61.62
37.72
Cl
Cl
OH
10000
50000
0
200
150
100
50
0
ppm (t1)

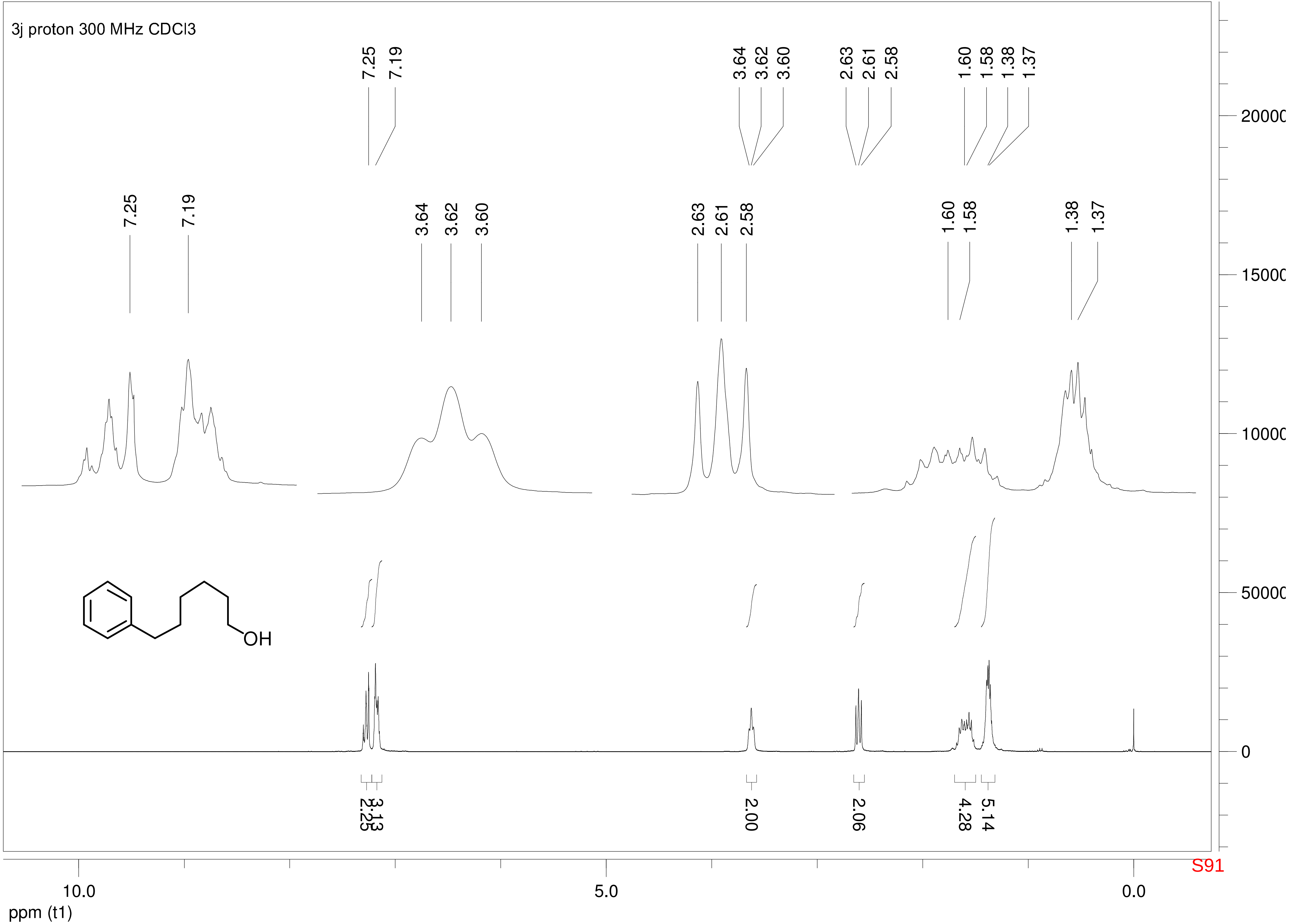

3i DEPT135 75 MHz CDCl3
129.43
128.84
127.16
61.68
37.78
Cl
Cl
OH
50000
0
200
150
100
50
0
ppm (t1)



3i proton 300 MHz CDCl3
7.37 7.36 7.35 7.34 7.26 7.21 7.20 7.19 7.18 7.17 7.14 7.12
3.92 3.90 3.88 3.86
3.08 3.05 3.03
1.56 1.54 1.52
1.00
1.01
1.13
2.19
2.17
1.10
Cl
Cl
OH
20000
15000
10000
50000
0
10.0
5.0
0.0
ppm (t1)


3j carbon 75 MHz CDCl3
142.69
128.35
128.20
125.57
77.42
77.20
77.00
76.58
62.93
35.84
32.64
31.42
29.02
25.56
OH
60000
50000
40000
30000
20000
10000
0
200
150
100
50
0
ppm (t1)


3j DEPT135 75 MHz CDCl3
128.42
128.28
125.64
63.01
35.91
32.71
31.50
29.10
25.64
OH
50000
0
200
150
100
50
0
ppm (t1)


3j proton 300 MHz CDCl3
7.25
7.19
3.64
3.62
3.60
2.63
2.61
2.58
1.60
1.58
1.38
1.37
OH
2.25
3.13
2.00
2.06
4.28
5.14
20000
15000
10000
5000
0
10.0
5.0
0.0
ppm (t1)