Internal Technical Report

Direct Digital Synthesizer based Clock/Synchronization Signal Generation Circuit for the GMRT Correlator

Rabi Swain and D. Anish Roshi
Date: 24-12-99

1 Introduction

The FFT (Fast Fourier Transform) chips used for the GMRT correlator require 4 extra clock cycles for performing a 512 point FFT. The GMRT samplers operate at 32 MHz. Thus to maintain the data throughput, the FFT (and MAC – Multiplier and Accumulator) subsystem can be operated at 32.25 MHz ($= 32$ MHz $\times$ 516/512). In this scheme, the delay-DPC subsystem takes the data from the sampler at 32 MHz and sends to the FFT subsystem at 32.25 MHz. For synchronizing the data flow at the two clock rates in the delay-DPC a synchronizing signal is required. This report describes a circuit which was built using two DDSs for the generation of 32 MHz and 32.25 MHz clocks and the required synchronization signal.

2 Requirement

For maintaining the data throughput the two subsystems operating at the two clocks (say f_1 and f_2) are required to be in the ratio 516/512 (= 129/128). The clock frequencies are related by the equation $f_1 = 512/516 \times f_2$. Since the maximum baseband bandwidth is 16 MHz f_1 should be 32 MHz for Nyquist sampling. Two active low signals of one clock cycle wide and same periodicity are to be generated from f_1 and f_2 for the synchronization of data flow in the delay-DPC. This can be achieved by taking the carry of Modulo 129 and Modulo 128 counters operated by clocks f_2 and f_1 respectively. The relative phase of these two carries should be same whenever the system is powered on or reset. The RMS time jitter of the sampling clock is required to be better than 200 ps.

3 Circuit

3.1 DDS

We used AD9851 direct digital synthesizers (DDS) for building the clock/synchronization signal generation circuit. The frequency resolution of the device is $f_{clk}/2^{32}$ and the phase resolution is $360^\circ/2^5 \doteq 11^\circ.25$. Here f_{clk} is the clock frequency of the DDS. We have chosen f_{clk} as 105 MHz, which will be taken from the local oscillator system of the GMRT. Since the phase of the clock be controlled in the DDS it is ideal for our application.

3.2 Block Diagram

The block diagram of the circuit is given in Fig 1. DDS1 and DDS2 are used to generate the two clocks f_1 and f_2. An EPLD (EPM7064) based circuit is used to preset the clocks to 32 and 32.25 MHz when powered on. It is also used to interface a computer to the DDS.

The outputs of the DDSs are passed through a 40 MHz low-pass filters. The filtered outputs are converted to TTL compatible square waves using the internal comparators in the DDSs. A modulo 128 and a modulo 129 counters, which are implemented in PAL 22V10 devices, are clocked by the 32 and 32.25 MHz outputs respectively. A circuit (PALCE16V8) enables these two counters twenty three f_{clk} cycles after the DDSs are programmed (i.e after the signal FQ_UD is activated). To check the frequencies f_1 and f_2 are in the ratio 128/129, a feed back circuit is implemented using an XOR logic. On the generation of the feed back signal the frequency words which are present in the internal registers of the DDSs are reloaded. This feed back circuit also generates an interrupt to the PC and a signal which can be read by the PC through the parallel port. The interrupt signal is several msec wide. The signal which is send to PC through the parallel port stays active till the computer sends an acknowledgment.

3.3 Circuit Diagram

The circuit diagram for the DDS based clock/synchronization signal generator is shown in Fig. 2. Fig. 3 shows the schematic of the circuit implemented in the EPLD EPM7064. It has two functions. (a) To preset the DDSs to frequencies 32 and 32.25 MHz when powered on or reset from computer. The signal timings and the frequency words loaded to the DDSs at this stage are given in Fig. 4. (b) After presetting the DDSs the EPLD gives the control (control signal *enable*) to the computer and provides the necessary interfacing circuits. The simulation results of the EPLD circuit is given in Fig. 5.

The modulo 128 and 129 counters and the feed back circuit (carry_check) are implemented in PAL22V10s. The synchronization of these PALs and the clock for the EPLD are generated using two PALCE16V8s. The programs of PAL22V10s and PALCE16V8s are also attached.

4 Programming the DDS

The DDS can be programmed to a desired frequency f by loading a suitable frequency word. A sample program is attached. As described above a feed back circuit checks whether the frequencies are in the ratio 128/129. In applications where the ratio is not 128/129, the feed back circuit can be disabled by sending the control word 0xCC and pulling the comp_reset signal low. The frequency words can then be loaded as given in the sample program. The other control words and their functions are : (a) 0xAA + comp_reset low – for resetting and loading the default frequencies to the DDSs; (b) 0xAE + comp_reset low – to reset the DDSs alone.

5 Test Results

5.1 Spectrum of the Output signals

Fig. 6 to Fig. 14 show the spectra of the 105 MHz clock used for testing and the two outputs of the DDSs. The 105 MHz input to the system is fed from Hp (Hewlett Packard) synthesized signal generator 8644A and all spectra are measured using the Hp spectrum analyzer 8591E. The phase noise of the outputs of DDS is ~ -70 dBc at 1 KHz offset. The power of the sidebands due to "intermodulation" of the two outputs are -53 dBc and -39 dBc respectively for 32.25 MHz and 32 MHz signals. All other spurious signals over a span of 20 MHz about the fundamental are $>$ 40 dBc for both outputs

5.2 Clock Jitter

Differential jitter of the rising edge of the output clocks are measured using a 400 MHz Tektronix oscilloscope 2465B. The jitter is measured about 9.6 μsec after triggering with the same clock and the time resolution used is 500 ps/div. The measured jitter is about 500 ps, which is comparable with the jitter of the 105 MHz clock measured using the same test setup. Thus we feel that the jitter measurements are limited by the measuring instrument and should be better than 350 ps ($= 500/\sqrt{2}$).

5.3 Synchronization signal

The synchronization signals generated by 32 MHz and 32.25 MHz are aligned with in an accuracy of 9.5 nsec ($1/105 \times 10^{-6}$ sec). The specified accuracy in nsec is the relative delay between the two synchronizing signals generated by the DDS output clocks. For the correlator application this accuracy is sufficient. The measurement is done with the 400 MHz oscilloscope 2465B.

5.4 Programming the DDS

The two DDSs can be programmed to a specified frequency using the software described in Section 4. The highest frequency that can be set is limited to $105/3 = 35$ MHz. The programmed frequencies at the two outputs are measured with Hp frequency counter 5386A.

6 More Work

We list a set of tasks that has to be done on the DDS system.

1. A better heat sink for the regulators.
2. The power on reset circuit is not functioning. **So a computer reset to the DDS system is required after power on.**
3. The feed back circuit is not tested.

4. The opto-isolators are presently bypassed. This means that there is no ground isolation between computer ground and DDS system ground. (Our feeling is that this may not be required.)

7 Acknowledgment

We thank Ganesh Jangam, M. Shaikh Jafar Ahmad, Mahesh Burse and Bhelrao in the correlator laboratory and C. M. Venkatesh in work shop for helping us to make the DDS based design concept a reality. We also thank S. A. Nipane for helping in modifying the GPIB interface program.

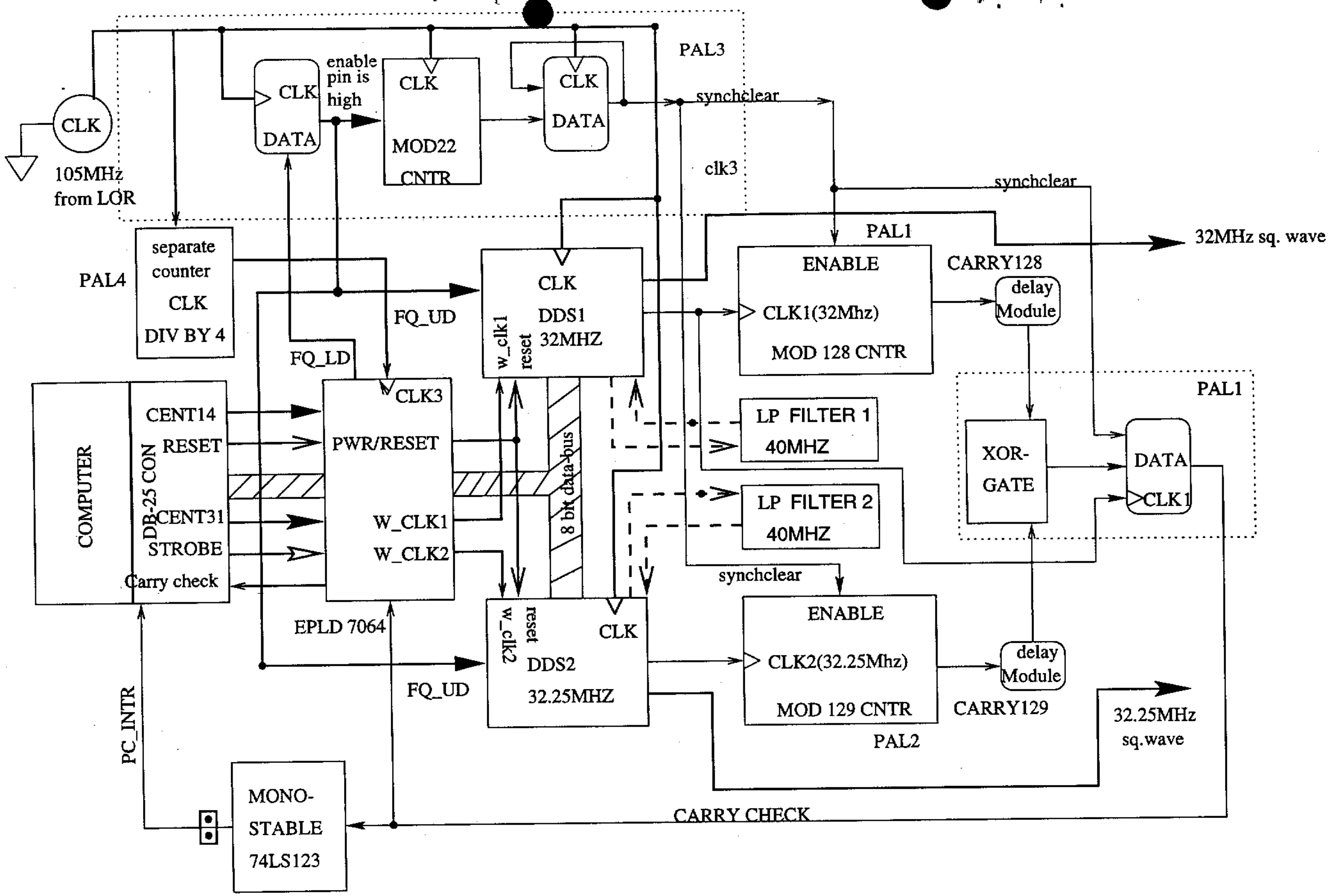
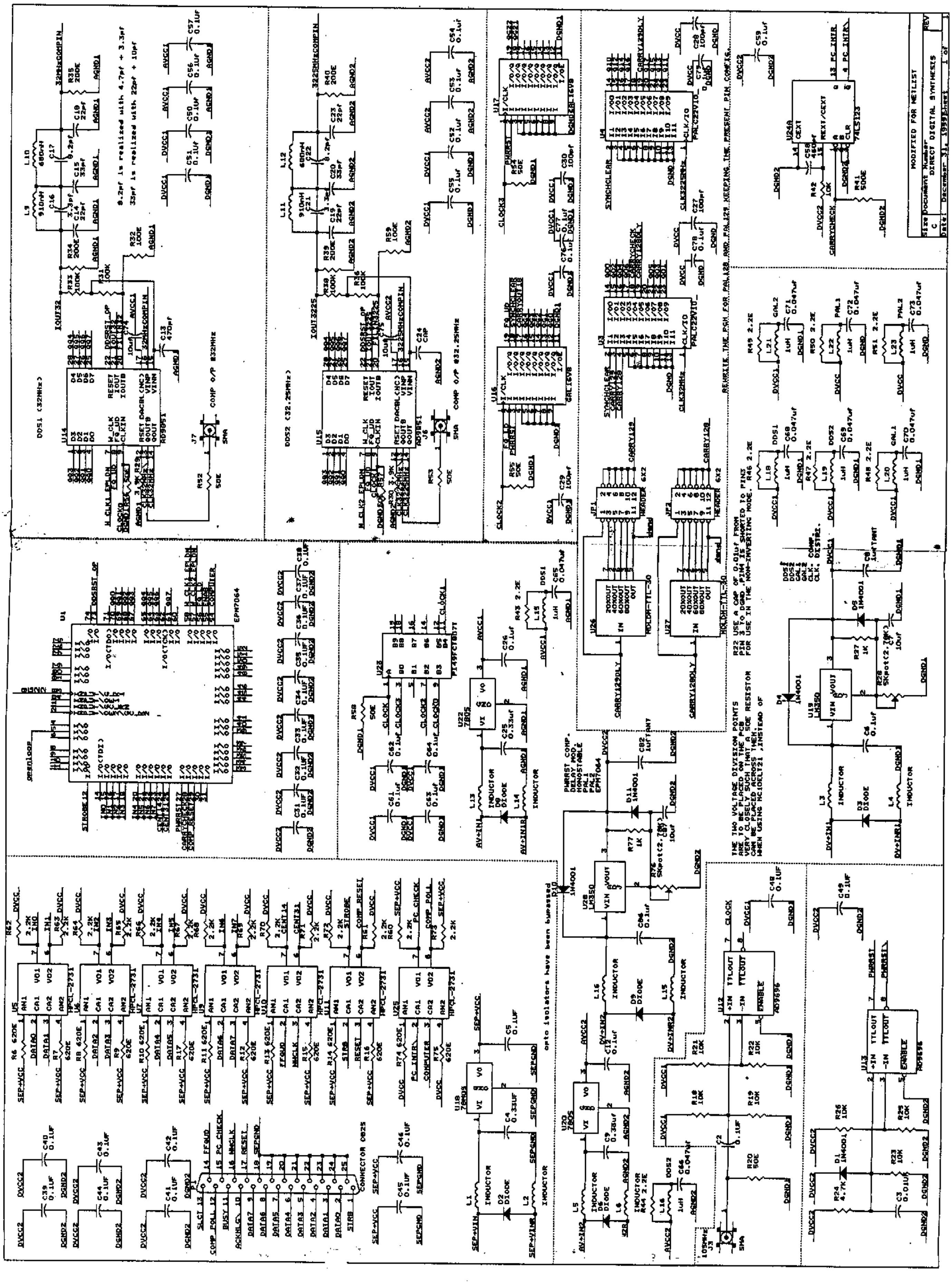


Fig 1. **Clock and Synchronization Signal Generator for the GMRT Correlator**

Fig 2



DDS1 (32MHz)
DDS2 (32.25MHz)
AD9851
EPM7064
GAL16V8
PALCE22V10
HDLDM-TTL-20
74LS123
LM350
7805
AD9696
HPCL-2731
CONNECTOR DB25
COMP O/P @32MHz
COMP O/P @32.25MHz
8.2pf is realized with 4.7pf + 3.3pf
33pf is realized with 22pf + 10pf
REWRITE THE PGM FOR PAL128 AND PAL129 KEEPING THE PRESENT PIN CONFIG.
R22 USE A CAP OF 0.01uf FROM PIN 3 TO GND PIN 2 IS SHORTED TO PIN 3 FOR USE IN THE NON-INVERTING MODE.
THE TWO VOLTAGE DIVISION POINTS ARE TO BE PLACED ON THE PCB VERY CLOSELY SUCH THAT A 50E RESISTOR CAN BE PLACED ACROSS THEM WHEN USING MC10ELT21 INSTEAD OF
opto isolaters have been bypassed
MODIFIED FOR NETLIST
Size C
Document Number
DIRECT DIGITAL SYNTHESIS
Date: December 31, 1999
Sheet 1 of 1
REV

MAX+plus II 9.21 File: FREQ1.GDF Date: 01/03/2000 15:59:22 Page: 1

Fig 3

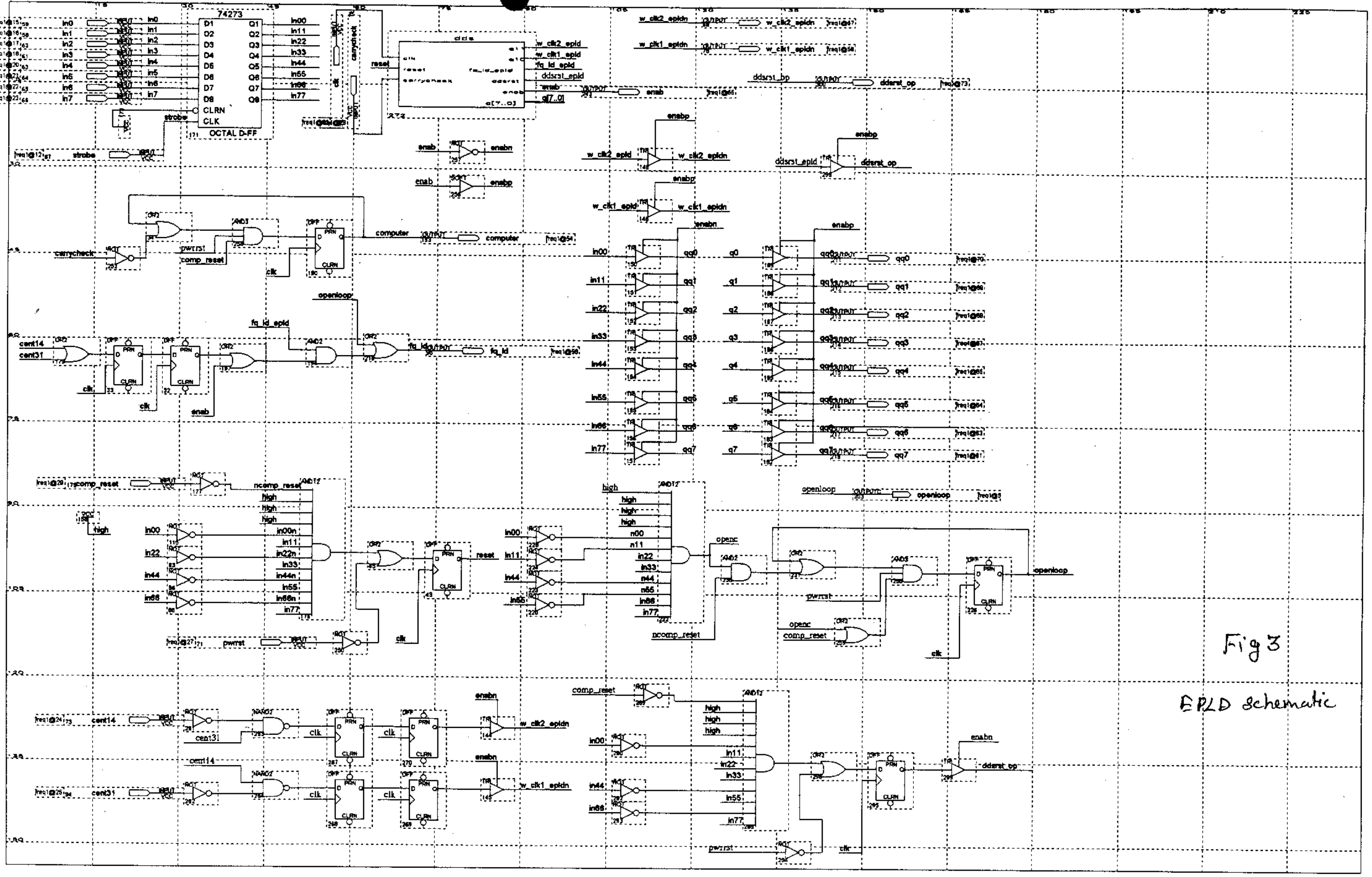


Fig 3

EPLD schematic

MAX+plus II 9.21 File: E:\HOME\SWAIN\MAXWORK\DDS\DDS.WDF Date: 01/03/2000 15:56:16 Page: 1

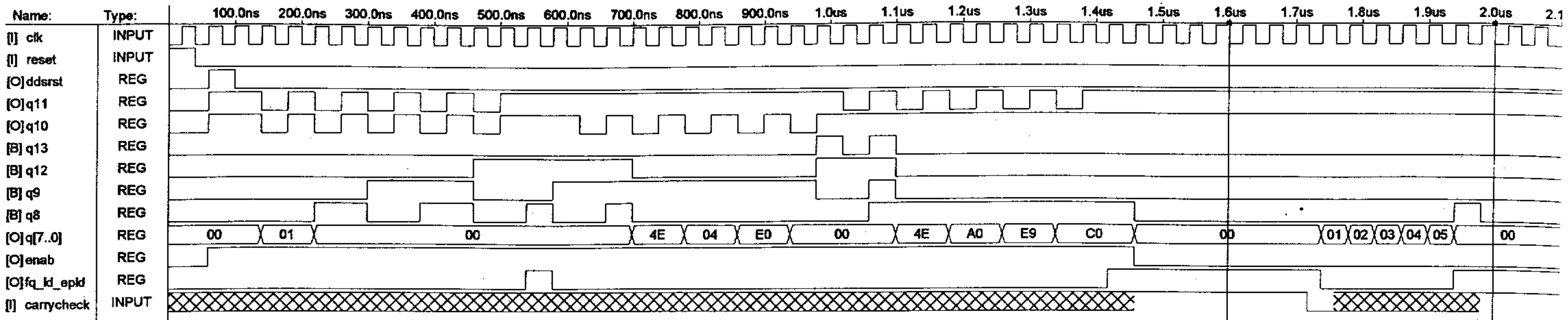


Fig 4. Timing diagram of DDS block in Fig3

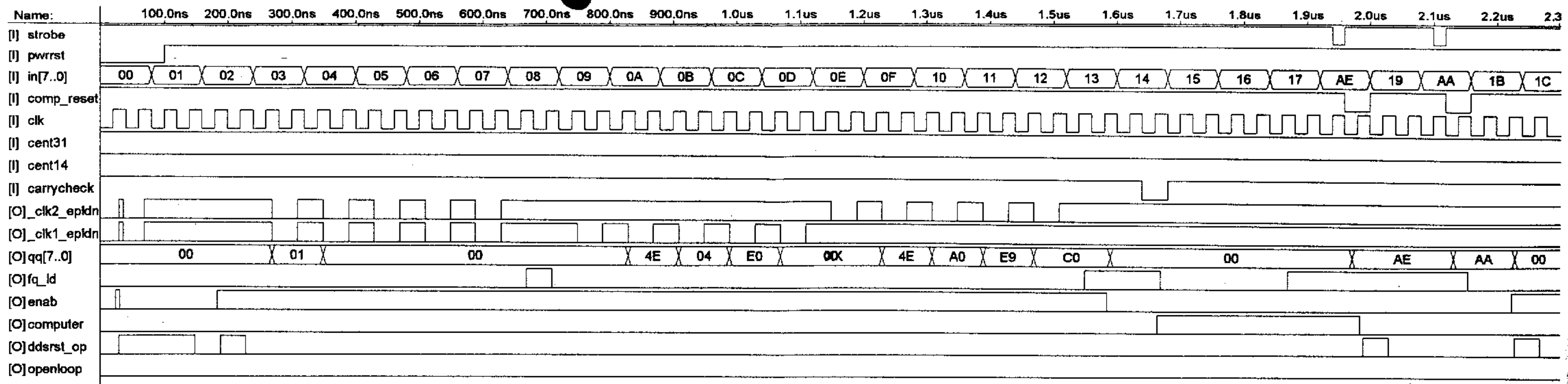


Fig 5: Simulation output of the schematic shown in Fig 3

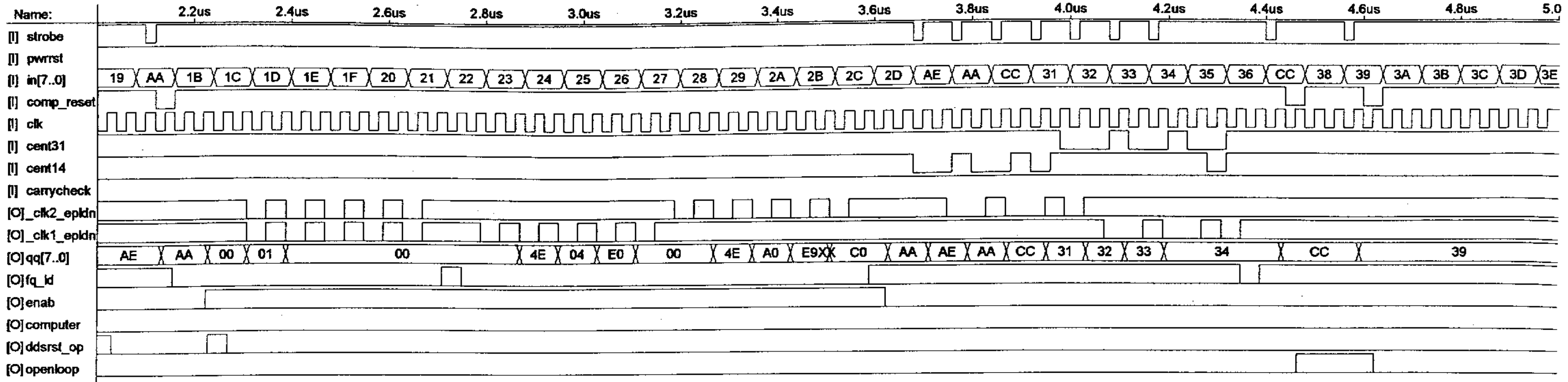


Fig 5 (Continued)

8644A]

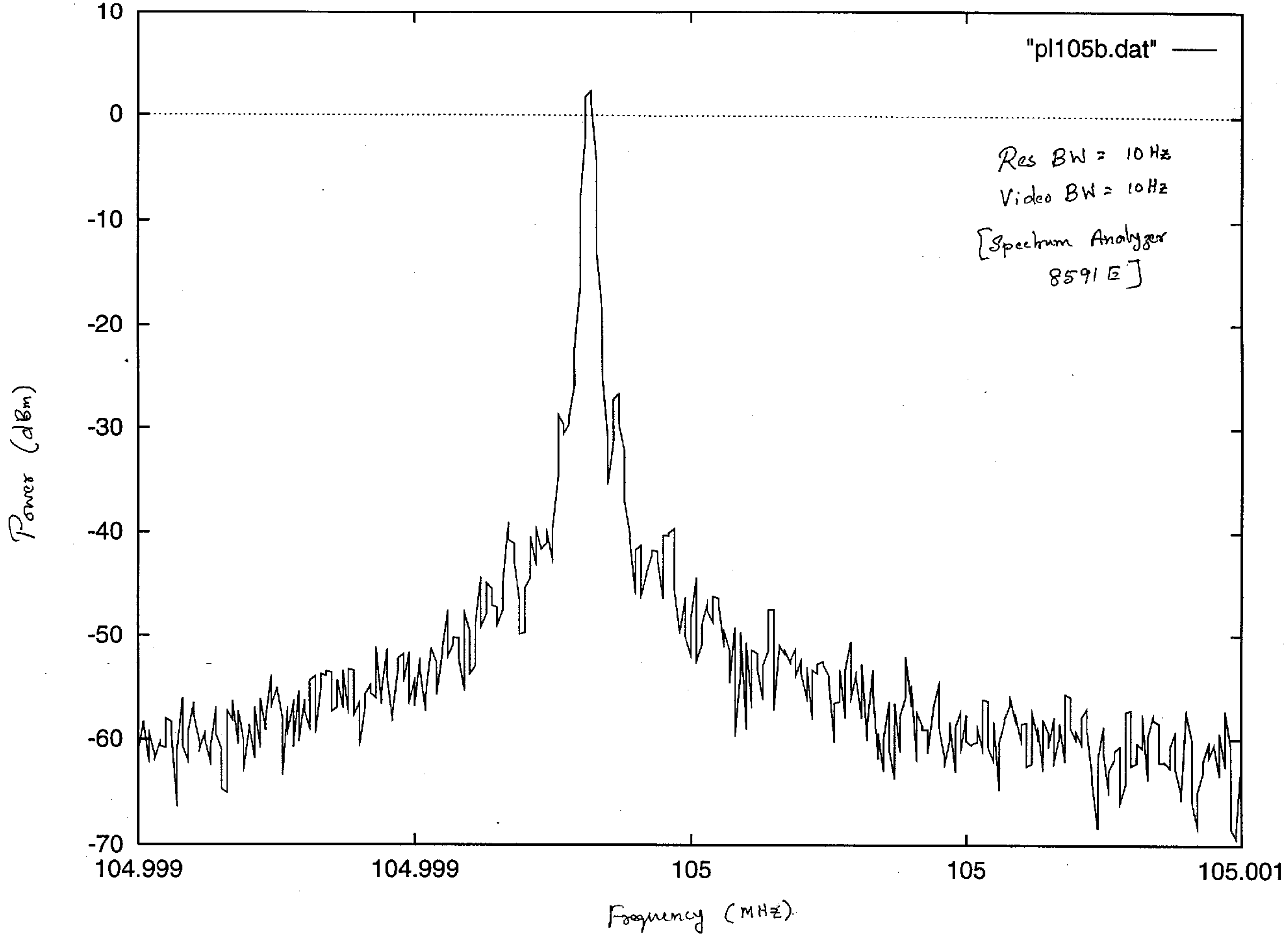
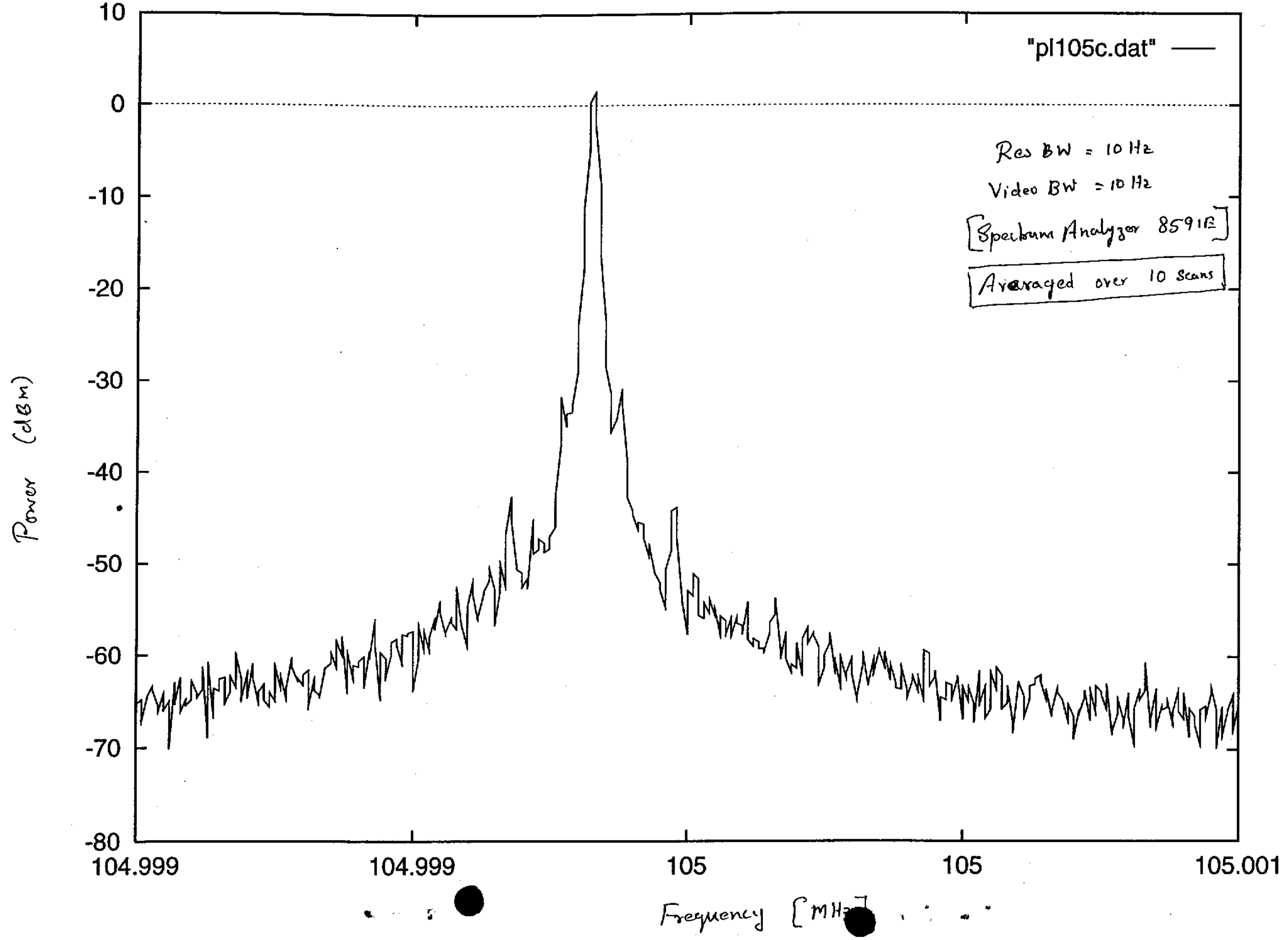
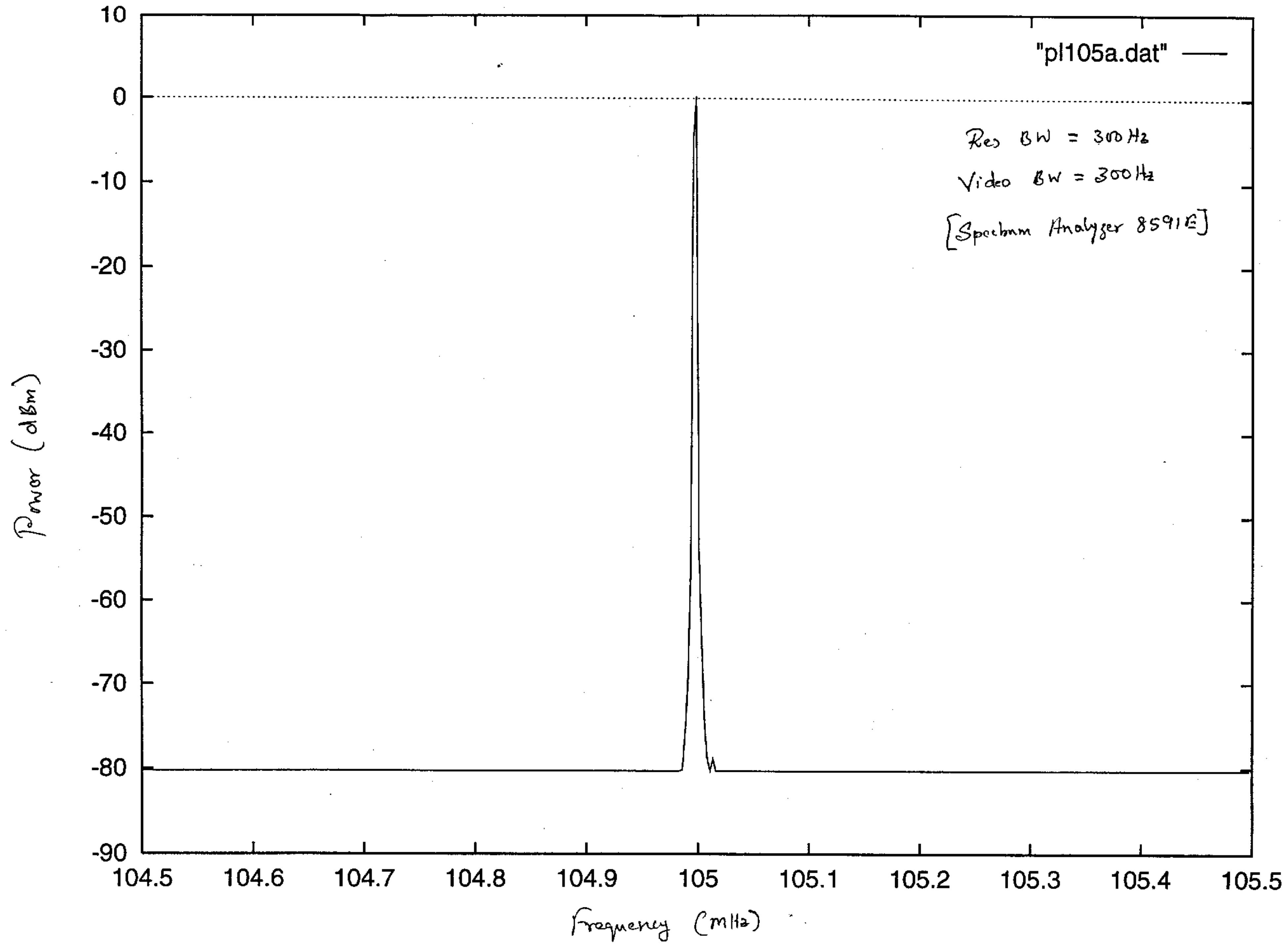

"pl105b.dat"
Res BW = 10 Hz
Video BW = 10 Hz
[Spectrum Analyzer
8591 E]
Power (dBm)
Frequency (MHz)
10
0
-10
-20
-30
-40
-50
-60
-70
104.999
104.999
105
105
105.001

Fig 7. [illegible]

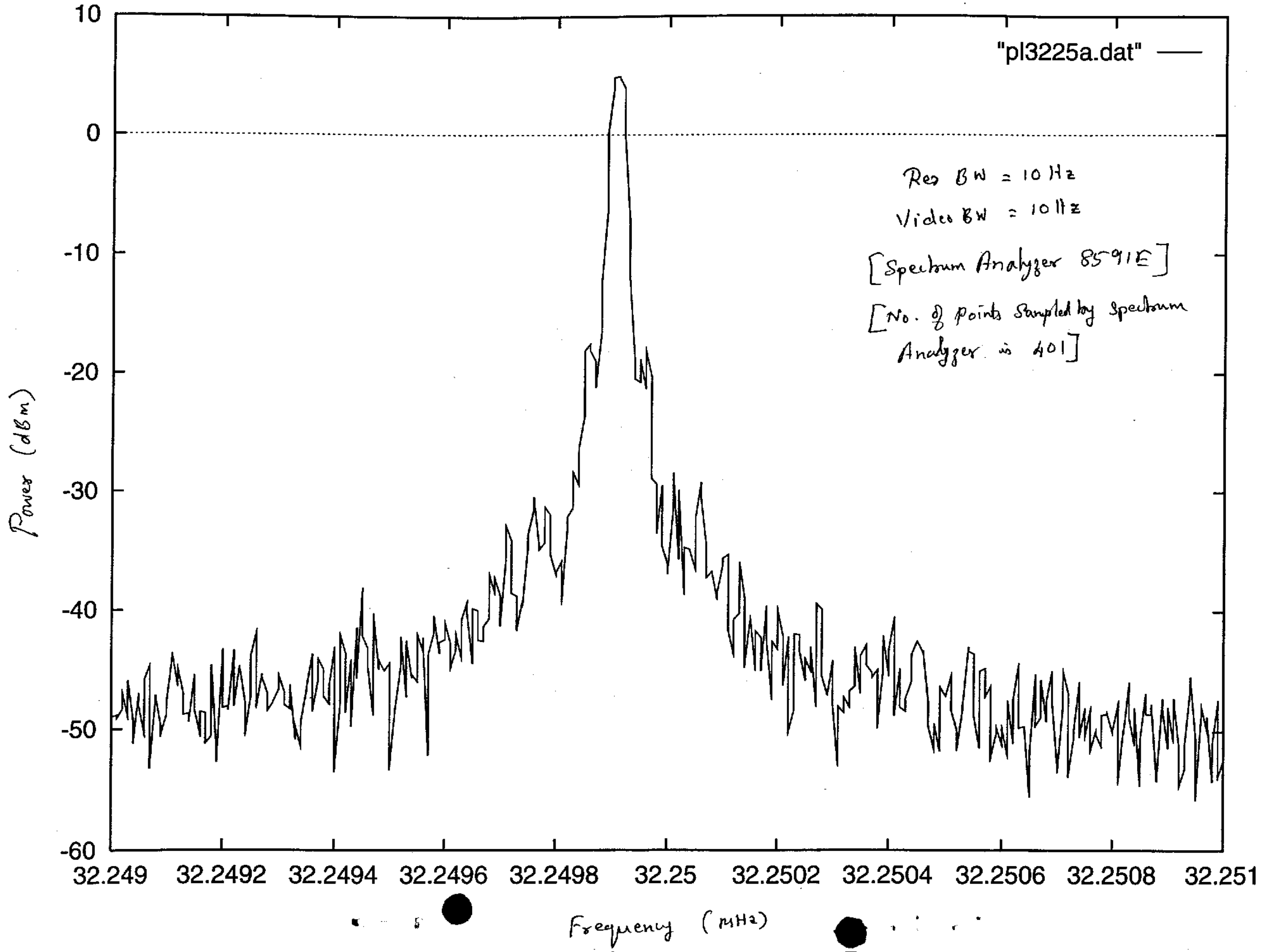


"pl105c.dat"
Res BW = 10 Hz
Video BW = 10 Hz
[Spectrum Analyzer 8591E]
Avaraged over 10 scans
Power (dBm)
Frequency [MHz]
10
0
-10
-20
-30
-40
-50
-60
-70
-80
104.999
104.999
105
105
105.001

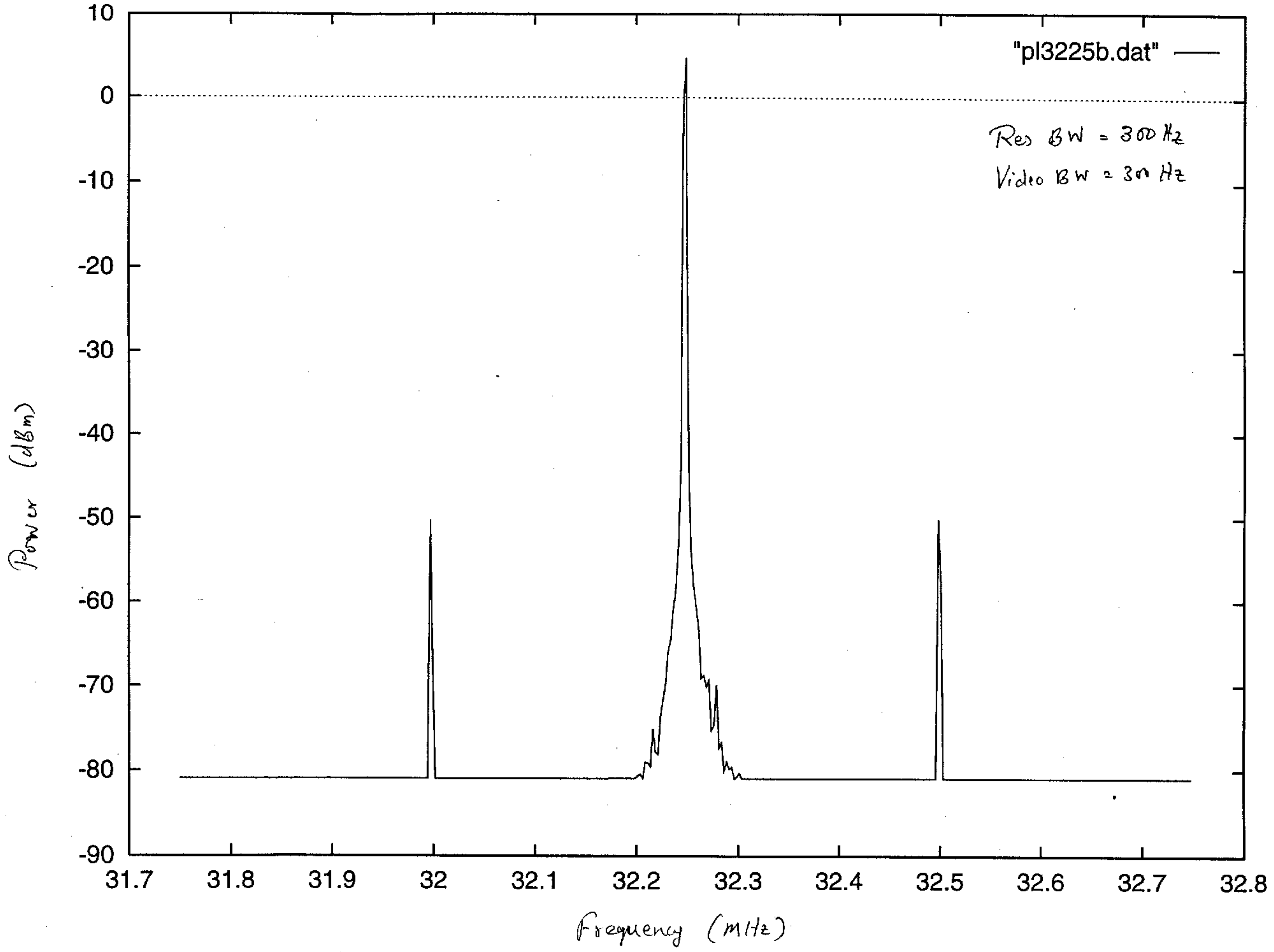
[illegible]


"pl105a.dat"
Res BW = 300 Hz
Video BW = 300 Hz
[Spectrum Analyzer 8591E]
Power (dBm)
Frequency (MHz)
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
104.5
104.6
104.7
104.8
104.9
105
105.1
105.2
105.3
105.4
105.5

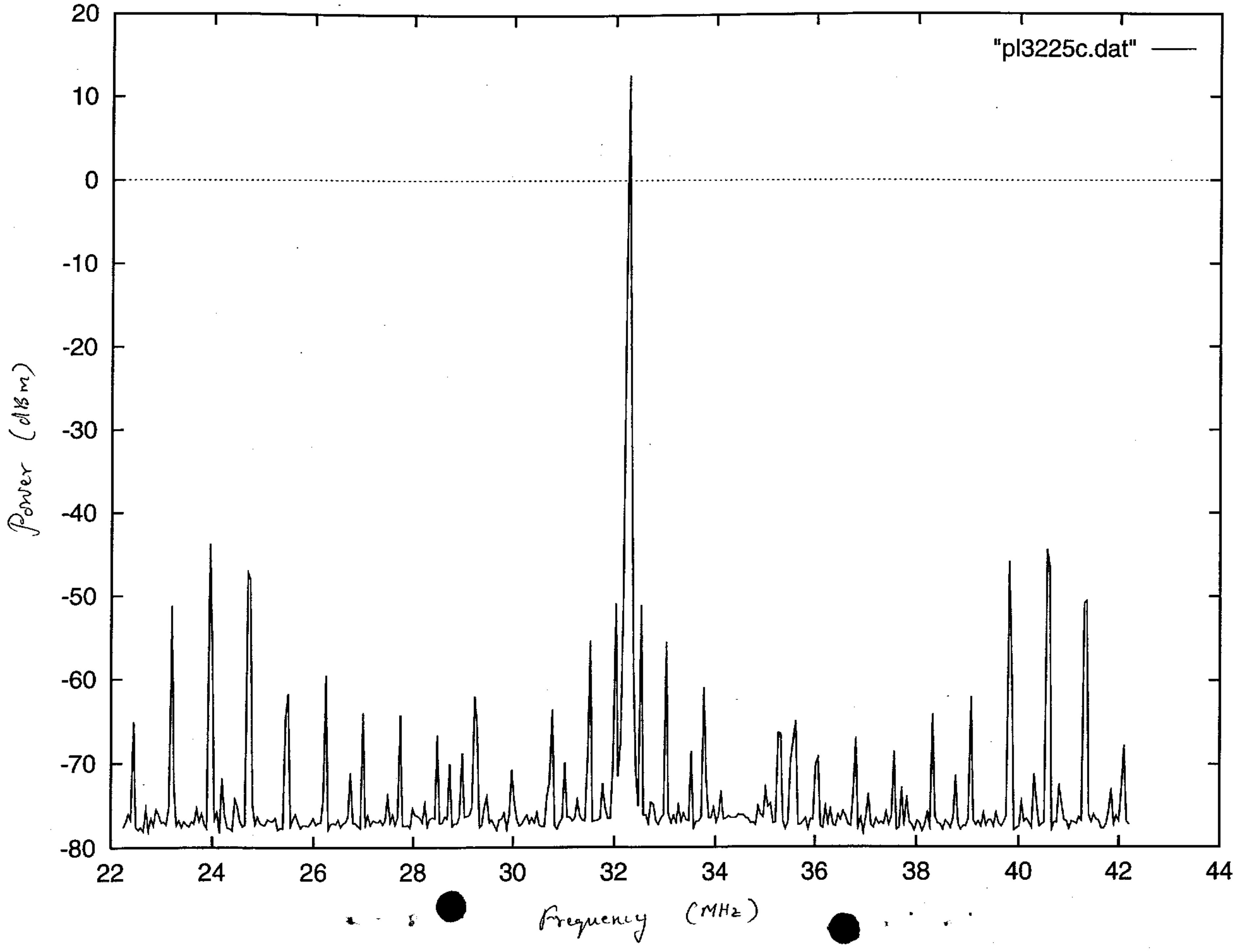
[illegible]



Res BW = 10 Hz
Video BW = 10 Hz
[Spectrum Analyzer 8591E]
[No. of points Sampled by Spectrum Analyzer is 401]

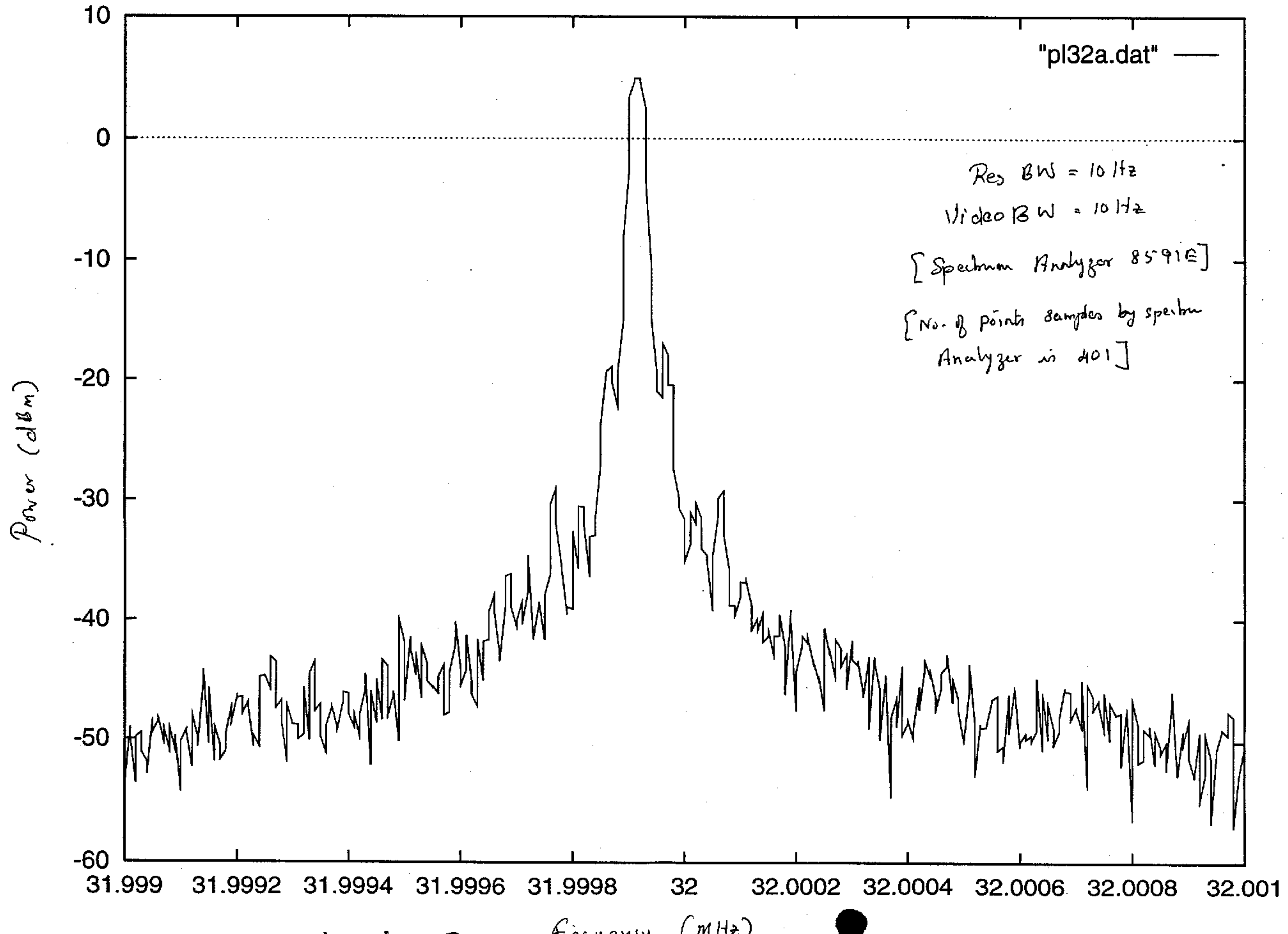

"pl3225b.dat"
Res BW = 300 Hz
Video BW = 300 Hz
Power (dBm)
Frequency (MHz)
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
31.7
31.8
31.9
32
32.1
32.2
32.3
32.4
32.5
32.6
32.7
32.8

[illegible]



"pl3225c.dat"
20
10
0
-10
-20
-30
-40
-50
-60
-70
-80
22
24
26
28
30
32
34
36
38
40
42
44
Power (dBm)
Frequency (MHz)

Fig 12: 32 MHz DDS o/p



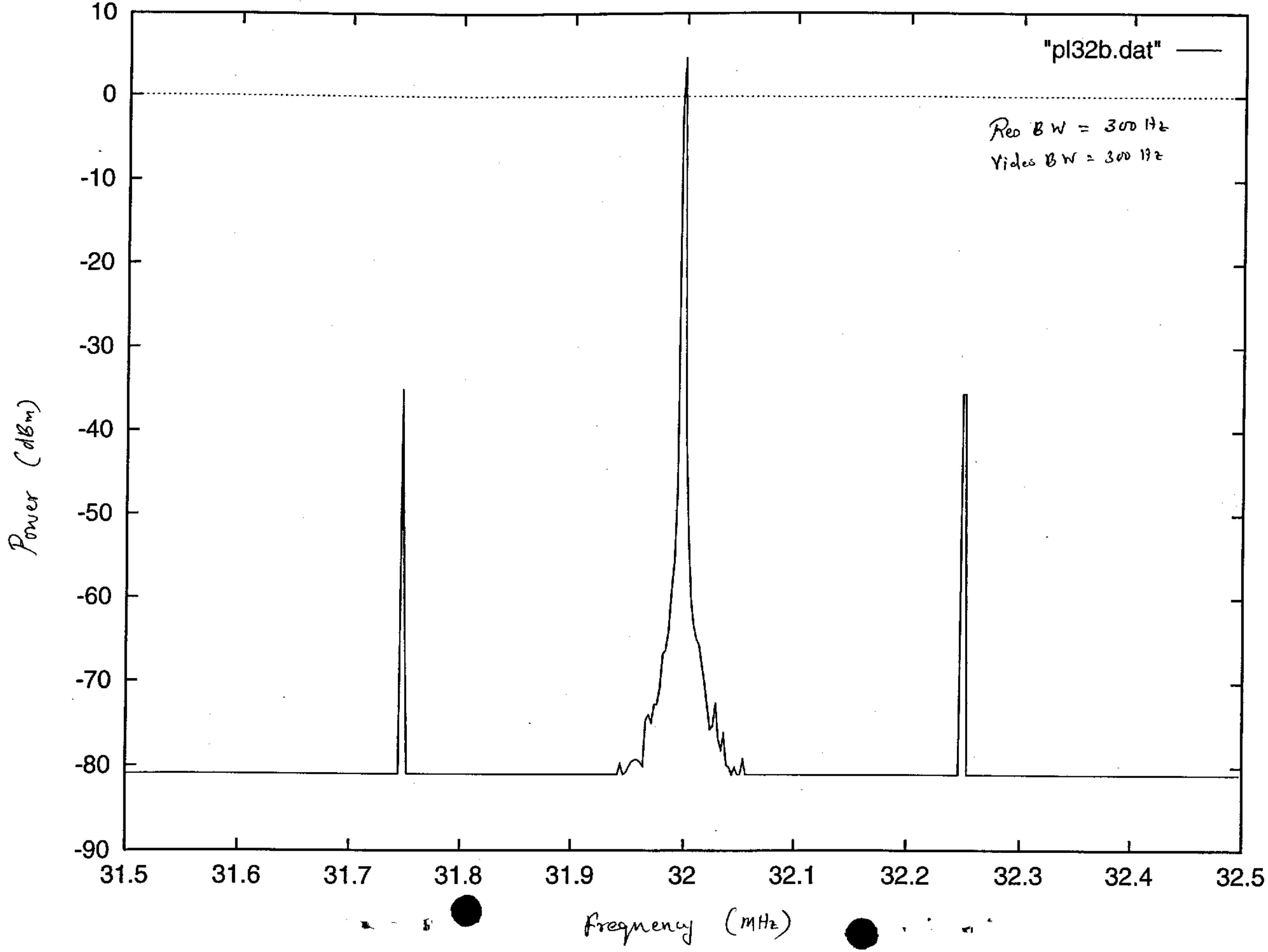
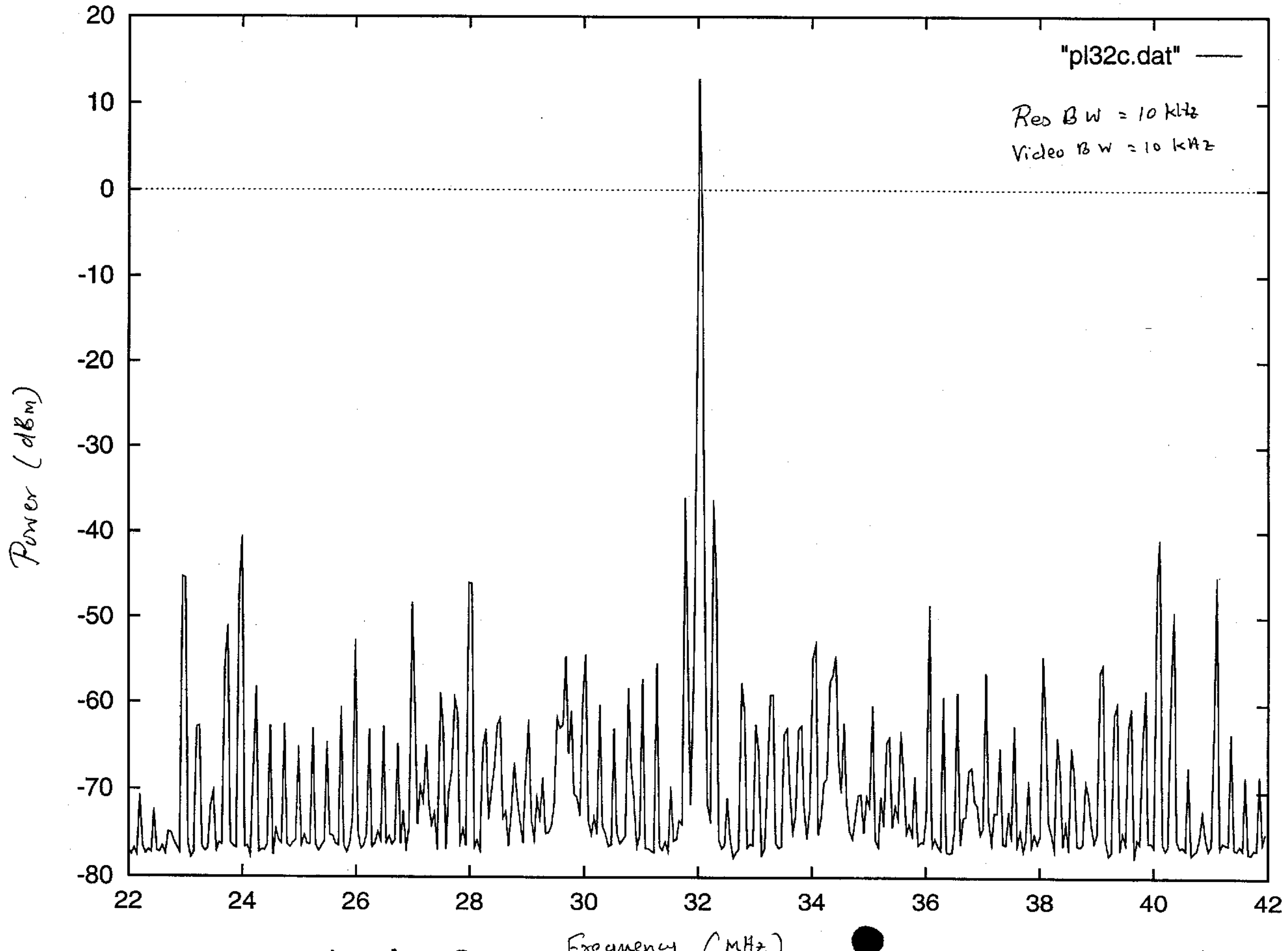

"pl32b.dat"
Res BW = 300 Hz
Video BW = 300 Hz
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
31.5
31.6
31.7
31.8
31.9
32
32.1
32.2
32.3
32.4
32.5
Power (dBm)
Frequency (MHz)

Fig 14: 32 MHz o/p from DDS



PAL PROGRAMS USED IN THE DDS SYSTEM

```
module counter23
title 'PAL16V8R
       NCRA-TIFR
       GAL-1'
       count23 device 'p16v8r'      ;
CLK                           PIN 1 ;
FQ_LD                         PIN 2 ;
PWRRST                        PIN 3 ;
OC                            PIN 11;

Q0                            PIN 12;
Q1                            PIN 13;
Q2                            PIN 14;
Q3                            PIN 15;
Q4                            PIN 16;
CARRYOUT22                    PIN 17;
SYNCHCLEAR                    PIN 18;
FQ_UD                         PIN 19;

H,L,X,Z,C = 1,0,.X.,.Z.,.C. ;
COUNTBY22 = [Q4,Q3,Q2,Q1,Q0] ;

Q0,Q1,Q2,Q3,Q4,SYNCHCLEAR,CARRYOUT22,FQ_UD   istype 'invert' ;

EQUATIONS
                FQ_UD.C = CLK ;
            COUNTBY22.C = CLK ;
           CARRYOUT22.C = CLK ;
           SYNCHCLEAR.C = CLK ;
               FQ_UD.OE = !OC ;
           COUNTBY22.OE = !OC ;
          CARRYOUT22.OE = !OC ;
          SYNCHCLEAR.OE = !OC ;

                FQ_UD := FQ_LD & PWRRST ;

            COUNTBY22 := (COUNTBY22+1)&(PWRRST)&(FQ_UD) ;

           CARRYOUT22 := (COUNTBY22==^h15) & (PWRRST)    ;

           SYNCHCLEAR := (SYNCHCLEAR # CARRYOUT22)&(FQ_UD)&(PWRRST);

end counter23 ;
```

```
module counter4
```

```
title 'PAL16R8
       schmatic name-U17
       GAL-2'
       count4 device 'p16v8r'     ;
CLK                            PIN 1 ;
OC                             PIN 11;

Q11                            PIN 18;
Q12                            PIN 19;

H,L,X,Z,C = 1,0,.X.,.Z.,.C. ;
COUNTBY4  = [Q12,Q11] ;

Q11,Q12   istype 'invert' ;

EQUATIONS

          COUNTBY4.C = CLK ;
          COUNTBY4.OE = !OC ;
          COUNTBY4 := (COUNTBY4 + 1);
TEST_VECTORS

end counter4 ;

-----------------------------------------------------


module mod128count
title 'mod 128 counter
       pal-1'

       mod128 device 'p22v10' ;

CLK32MHz                 PIN 1 ;
SYNCHCLEAR               PIN 2 ;
CARRY129I                PIN 3 ;
CARRY128I                PIN 4 ;
OC                       PIN 5 ;

Q00                      PIN 14 ;
Q01                      PIN 23 ;
Q02                      PIN 15 ;
Q03                      PIN 22 ;
Q04                      PIN 16 ;
Q05                      PIN 21 ;
Q06                      PIN 17 ;

CARRY128O                PIN 19 ;
CARRYCHECK               PIN 18 ;

Q00,Q01,Q02,Q03,Q04,Q05,Q06   ISTYPE 'BUFFER';
CARRY128O, CARRYCHECK         ISTYPE 'INVERT';
```

```
H,L,C = 1,0,.C. ;
COUNT = [Q06,Q05,Q04,Q03,Q02,Q01,Q00] ;

EQUATIONS

            COUNT.C = CLK32MHz ;
           COUNT.OE = !OC ;
        CARRY128O.C = CLK32MHz ;
       CARRY128O.OE = !OC ;
       CARRYCHECK.C = CLK32MHz;
      CARRYCHECK.OE = !OC;

             COUNT := (COUNT + 1)&(SYNCHCLEAR);
        !CARRY128O := (COUNT==^h7F)&(SYNCHCLEAR) ;

       !CARRYCHECK := (CARRY128I $ CARRY129I)&(SYNCHCLEAR);


END mod128count

---------------------------------------------------------------
module mod129count
title 'mod 129 counter
       pal-1'

       mod129 device 'p22v10' ;

CLK3225MHz             PIN 1 ;
SYNCHCLEAR             PIN 2 ;
OC                     PIN 3 ;

Q00                    PIN 14 ;
Q01                    PIN 23 ;
Q02                    PIN 15 ;
Q03                    PIN 22 ;
Q04                    PIN 16 ;
Q05                    PIN 21 ;
Q06                    PIN 17 ;
Q07                    PIN 20 ;

CARRY129               PIN 19 ;

Q00,Q01,Q02,Q03,Q04,Q05,Q06,Q07 ISTYPE 'BUFFER';
CARRY129 ISTYPE 'INVERT';

H,L,C,X = 1,0,.C.,.X. ;
COUNT = [Q07,Q06,Q05,Q04,Q03,Q02,Q01,Q00] ;

EQUATIONS

            COUNT.C = CLK3225MHz ;
        CARRY129.C  = CLK3225MHz ;
           COUNT.OE = !OC ;
        CARRY129.OE = !OC ;

             COUNT := (COUNT + 1)&(SYNCHCLEAR)&(!(COUNT.FB>=^h80)) ;
         !CARRY129 := (COUNT==^h80)&(SYNCHCLEAR) ;

END mod129count ;
```

DDS Control Program

```
#include<stdio.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>

/* CONTROL BITS  --  0   0   0   0  ~RESET WWCLK ~FFQUD ~STRB
                     b7  b6  b5  b4  b3     b2      b1      b0   */
/* CONTROL WORD  --  0   0   0   0   0    ~pwr-dn con     con */

#define DATAPORT 0x378
#define CNTRLPORT 0x37A
#define RESETBYTE 0xAA
#define DDSRSTBYTE 0xAE
#define OPENLBYTE 0xCC
#define HIGH 1
#define LOW  0
#define DEFAULT37A 0x04
#define DDSCLOCK 105  /* MHz */

int FQLD(int STATE)
{
   switch(STATE)
   {
         case 0:
           outportb(CNTRLPORT, ((DEFAULT37A | 0x02)&0x0b));
           delay(1);
           break;
         case 1:
           outportb(CNTRLPORT, DEFAULT37A);
           delay(1);
           break;
    }
 }

 int init()
 {
    outportb(DATAPORT,0x00);
    outportb(CNTRLPORT,DEFAULT37A);
 }

 void STROBE()
 {
   outportb(CNTRLPORT,(unsigned char)(DEFAULT37A|1));
   delay(1);
   outportb(CNTRLPORT,(unsigned char)DEFAULT37A);
   delay(1);
 }

 void WCLK(int PORTID)
 {
   switch(PORTID)
   {
         case 0:
            outportb(CNTRLPORT,(unsigned char)(DEFAULT37A & 0x0b));
            delay(1);
            outportb(CNTRLPORT,(unsigned char)DEFAULT37A);
            delay(1);
            break;

          case 1:
            outportb(CNTRLPORT,(unsigned char)(DEFAULT37A | 0x02));
            delay(1);
            outportb(CNTRLPORT,(unsigned char)DEFAULT37A);
            delay(1);
            break;
    }
  }

  int send_freq(unsigned long FREQ_WD, int PORTID)
  {
    unsigned char SEND_BYTE;
    int i=0;

    while(i<5)
    {
          if(i==0)
           SEND_BYTE=0;
      else
           SEND_BYTE=(unsigned char)((FREQ_WD >> ((4-i)*8)) & 0x000000ff);

          outportb(DATAPORT,SEND_BYTE);
      STROBE();
          WCLK(PORTID);
          i++;
    }
  }

  int RESET()
  {
    outportb(DATAPORT,RESETBYTE);
    STROBE();
    outportb(CNTRLPORT, (unsigned char)(DEFAULT37A | 0x08));
    delay(1);
    outportb(CNTRLPORT, DEFAULT37A);
    outportb(DATAPORT,0x00);
    STROBE();
    delay(1);
  }

  int DDSRST()
  {
    outportb(DATAPORT,DDSRSTBYTE);
    STROBE();
    outportb(CNTRLPORT, (unsigned char)(DEFAULT37A | 0x08));
    delay(1);
    outportb(CNTRLPORT, DEFAULT37A);
```

```
   outportb(DATAPORT,0x00);
   STROBE();
   delay(1);
 }

 int LOCKPLL()
 {
    send_freq(0x01,0);
    send_freq(0x01,1);
    FQLD(LOW);
    FQLD(HIGH);
 }

 int OPENLOOP(int STATUS)
 {
   /* STATUS = 1 to open the loop; 0 to close */

   switch(STATUS)
   {
        case 0 :
           outportb(DATAPORT,0x00);
           break;
        case 1 :
           outportb(DATAPORT,OPENLBYTE);
           break;
        default :
           return 1;
   }

   STROBE();
   outportb(CNTRLPORT, (unsigned char)(DEFAULT37A | 0x08));
   delay(1);
   outportb(CNTRLPORT, DEFAULT37A);
   outportb(DATAPORT,0x00);
   STROBE();
   delay(1);
   return 0;
}

unsigned long int freqwd(double freq)
{
  unsigned long int FREQWD;

  if(freq > DDSCLOCK/3 || freq < 0)
  {
        printf("\n Freq out of range ");
        return 0;
  }

  FREQWD = (unsigned long int)(freq * pow(2,32)/ DDSCLOCK);
  printf("\n Freq = %7.4f and Freq word (hex) = %lx",freq,FREQWD);
  return FREQWD;
}
```

```
main(int argcnt, char *argv[])
{
        int mode;
        double freq1, freq2;
        unsigned long int FREQWD1, FREQWD2;

   init();

   if(argcnt < 3)
   {
        printf("\n Usage : setfreq freq1 freq2 mode");
        printf("\n mode = 0 to reset the system (default freqs) ");
        printf("\n mode = 1 to reset the DDS alone");
        printf("\n mode = 2 to open the loop ");
        printf("\n mode = 3 to close the loop ");
        printf("\n mode = 4 to set the frequencies and keep the loop op
en");
        printf("\n mode = 5 to set the frequencies using freq words (op
en loop) ");
        printf("\n Values of freq1 & freq2 are used only in mode 4 & 5"
);
        printf("\n In mode 4 the freq1 and freq2 are the freqs in MHz")
;
        printf("\n In mode 5 the freq1 and freq2 are the freq words in
hex");
        return;
   }

   mode = atoi(argv[3]);
   switch(mode)
   {
        case 0 :
               RESET();
               break;

        case 1 :
               DDSRST();
               break;

        case 2 :
               OPENLOOP(1);
               break;

        case 3 :
               OPENLOOP(0);
               break;

        case 4 :
          DDSRST();
          send_freq(0x01,0);
          send_freq(0x01,1);
          FQLD(LOW);
          OPENLOOP(0);
          FQLD(HIGH);
          OPENLOOP(1);
/*        LOCKPLL();    Lock the internal PPL in the DDS */
          sscanf(argv[1],"%lf",&freq1);
          sscanf(argv[2],"%lf",&freq2);
          FREQWD1 = freqwd(freq1);
          FREQWD2 = freqwd(freq2);
          send_freq(FREQWD1,0);
          send_freq(FREQWD2,1);
          OPENLOOP(0);
          FQLD(LOW);
          OPENLOOP(0);
          FQLD(HIGH);
          OPENLOOP(1);
          break;

        case 5 :
          DDSRST();
          send_freq(0x01,0);
          send_freq(0x01,1);
          FQLD(LOW);
          OPENLOOP(0);
          FQLD(HIGH);
          OPENLOOP(1);
/*        LOCKPLL();    Lock the internal PPL in the DDS */
          sscanf(argv[1],"%X",&FREQWD1);
          sscanf(argv[2],"%X",&FREQWD2);
          send_freq(FREQWD1,0);
          send_freq(FREQWD2,1);
          OPENLOOP(0);
          FQLD(LOW);
          OPENLOOP(0);
          FQLD(HIGH);
          OPENLOOP(1);
          break;

        default:
          printf("\n Invalid mode ");
   }

}
```