Virtual Crossover Software Manual

1 Introduction

This manual describes the program vc.exe, which is used to directly control and communicate with the Virtual Crossover. There are other programs related to the Virtual Crossover that are provided, for example to calculate the responses of circuits or of digital filters; these individual programs are not covered in this manual. Please see the documentation specific to these other programs for information about how to use them.

2 Installing libusb

You must have libusb-win32 installed on your computer to run vc.exe. Libusb-win32 is a freely available package of utilities for communications over the usb port.

There is more than one way to install the library and you are certainly welcome to read the accompanying documentation and install the library as you want. The following is a simple method that worked for me – simply double click on the file: libusb-win32-filter-bin-0.1.12.2.exe and accept the defaults suggested by the installer.

3 Installing the CC3250MT.DLL driver

When you first try to run vc.exe, you may get an error message saying that the cc3250mt.dll driver was not found. The compiler used for the vc.exe program (Borland C++ Builder) requires that this driver be present for the compiled program to run.

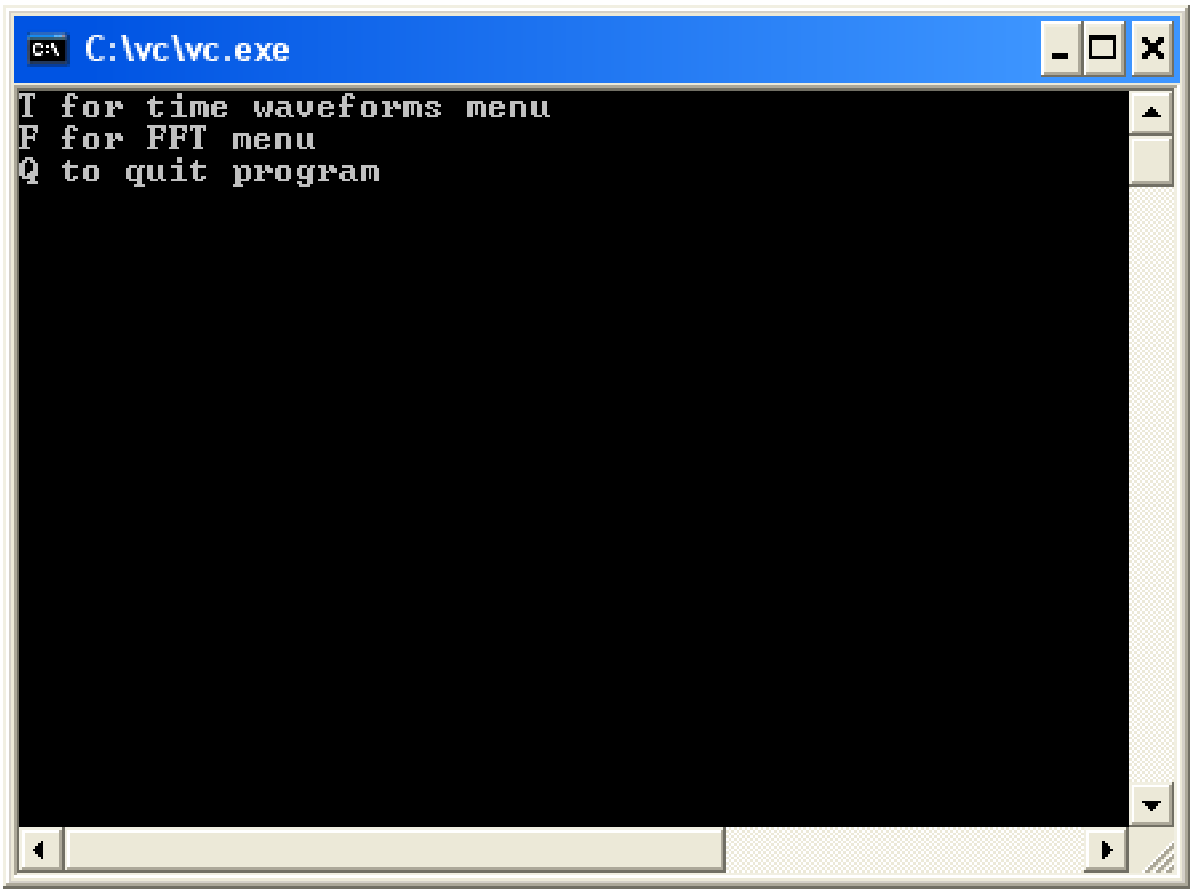
If you get this error, putting the cc3250mt.dll driver in your C :\WINDOWS\system32 folder should solve the problem.

4 Running The Program

To run the program, double click on vc.exe. The file vc.in must reside in the same directory as vc.exe and it must be in the proper format.

Options in the program are selected by typing letters corresponding to options listed in the program menus. The mouse cannot be used to select program options.

The program can be run in two basic modes, the first where the Virtual Crossover is attached to the computer running vc.exe over the usb port, and the second where the Virtual Crossover is not attached. If the second line in vc.in (the quantity vc_attached) is set to zero, the program does not attempt to find the Virtual Crossover on the usb bus, and it comes up in a restricted mode of operation. Here is the main menu screen that comes up in that case:

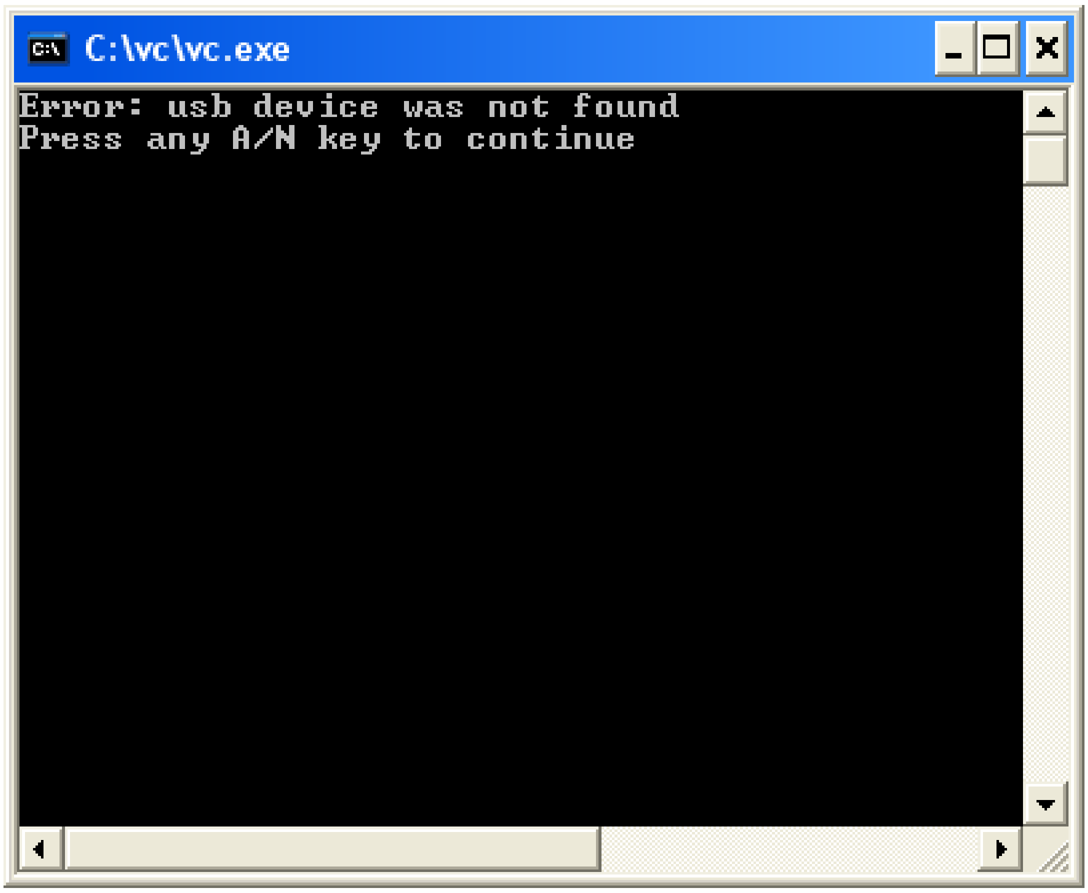


```
T for time waveforms menu
F for FFT menu
Q to quit program
```

Figure 1: Main menu with no Virtual Crossover attached.

Basically all you can do in this mode is read in waveforms in the time or frequency domain, display plots, carry out FFTs, and save waveforms. Any of the program options that involve communication with the Virtual Crossover cannot be accessed in this mode. Please see below for an explanation of the functions available in the Time Waveforms Menu and the FFT Menu.

On the other hand, if the second line of the vc.in file (vc_attached) is set to 1, the program vc.exe will attempt to locate the Virtual Crossover on the usb bus when it is invoked. If vc.exe is unable to find the Virtual Crossover, the following screen is displayed:

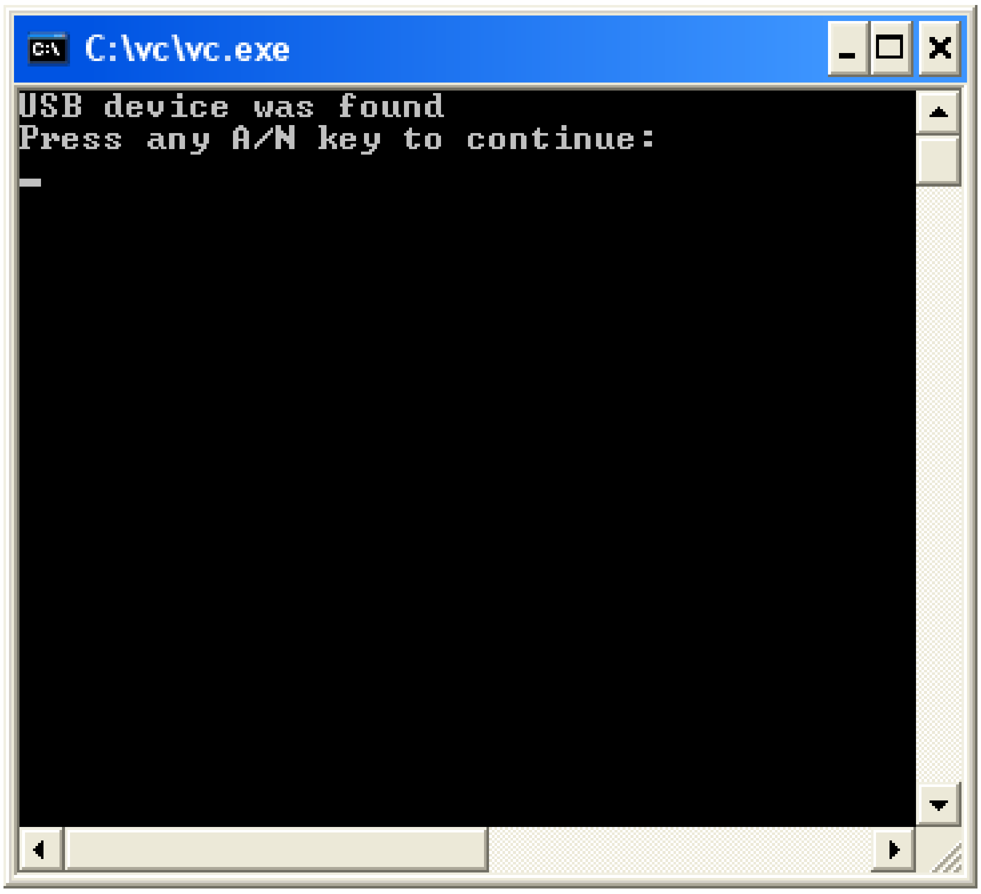


```
Error: usb device was not found
Press any A/N key to continue
```

Figure 2: Program is unable to locate the Virtual Crossover.

Typing any alpha/numeric key will terminate the program. If you see this screen, check that power is supplied to the Virtual Crossover (the front panel LED should be on), and that a suitable usb cable is connected between the computer and the Virtual Crossover. Also, libusb must be installed on your computer in order for vc.exe to be able to find the Virtual Crossover.

Assuming that vc.exe was able to locate the Virtual Crossover on the usb bus, the following screen will be displayed:



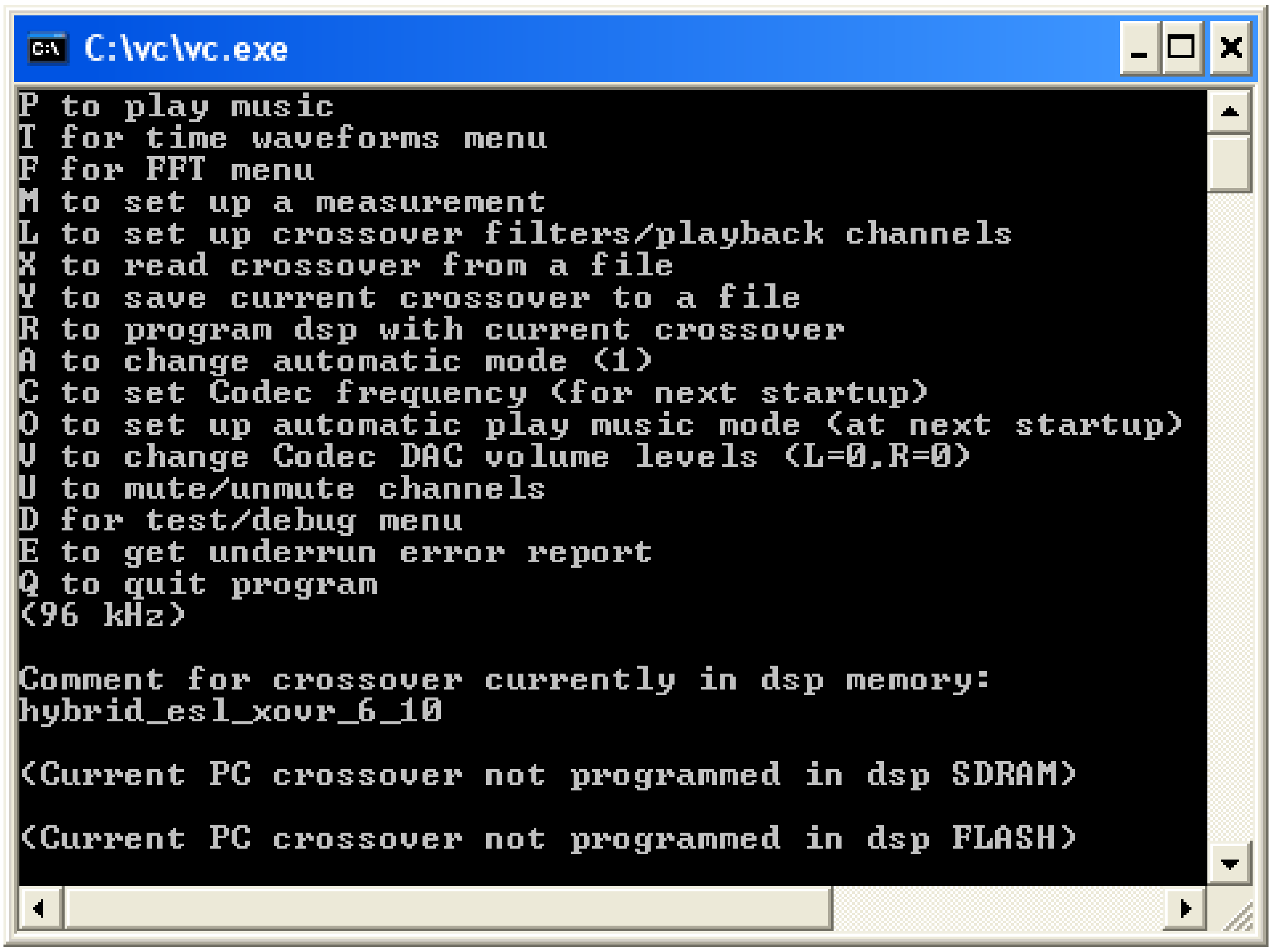
```
USB device was found
Press any A/N key to continue:
_
```

Figure 3: Screen displayed if vc.exe locates the Virtual Crossover.

At this point, typing any alpha/numeric key will bring up the Main Menu.

5 Main Menu

The following screen is displayed when you are in the Main Menu:



```
C:\vc\vc.exe
P to play music
T for time waveforms menu
F for FFT menu
M to set up a measurement
L to set up crossover filters/playback channels
X to read crossover from a file
Y to save current crossover to a file
R to program dsp with current crossover
A to change automatic mode (1)
C to set Codec frequency (for next startup)
O to set up automatic play music mode (at next startup)
V to change Codec DAC volume levels (L=0,R=0)
U to mute/unmute channels
D for test/debug menu
E to get underrun error report
Q to quit program
(96 kHz)

Comment for crossover currently in dsp memory:
hybrid_esl_xovr_6_10

(Current PC crossover not programmed in dsp SDRAM)

(Current PC crossover not programmed in dsp FLASH)
```

Figure 4: Virtual Crossover Main Menu display.

From the Main Menu, typing the single letter associated with each program option will access that function. For example, typing the letter t or T will access the Time Waveforms Menu (either lower or upper case letters will work). Before presenting explanations for each of the program sub-menus, a discussion regarding the crossover information displayed in the Main Menu will be given.

There are three different levels in which crossover information can reside in the Virtual Crossover and its associated software (i.e. in vc.ece). When the Virtual Crossover is powered up, it may or may not have crossover information already programmed in its FLASH memory. If the Virtual Crossover code (in vc.asm, which is the program embedded in the device itself) detects that a crossover has been programmed, it transfers the information to appropriate areas in its SDRAM memory, where it can have faster access to the crossover filters for processing waveforms. When you run vc.exe, the program checks the areas of dsp memory to see if there is a valid string of identifying text for a crossover that might already have been programmed in the dsp; if it finds a valid identifier for a crossover, it displays the information in the Main Menu. In Fig. 4, you can see the identifier "hybrid_esl_xovr_6_10" below the various sub-menu options; this is because I previously programmed this particular Virtual Crossover with a crossover of that name. Whenever you program the Virtual Crossover with a new crossover, you include an identifying string along with the numerical data, to help remind you of the particular crossover that is in memory.

Even though vc.exe knows the name of the crossover in dsp memory, it doesn't know the details of the crossover; for example, it doesn't have any information about the individual crossover filters that are in dsp memory. This is basically because the original crossover numerical data, consisting of time-domain filters, is transformed into the frequency domain before being programmed in the dsp. Therefore, whenever you set up a particular crossover that you want to program in the DSP, it is important to save the crossover to a file (option Y) as well as to program the crossover in the dsp (option R), so that you will be able to retrieve all the crossover data at a later time (option X, to read crossover data from a file).

As far as what is in dsp memory, the dsp uses the crossover that is in SDRAM memory. When the dsp first powers up, it checks if there is a crossover saved in FLASH memory and if so, it transfers the crossover to SDRAM memory for execution. If you have set up a crossover that you want to try but you are not sure yet if you want to permanently program it in FLASH memory, you can program it into SDRAM memory (see option R below) and the dsp will use the crossover for measurements or for playing music, as long as it is powered up. If you at some point decide that you want the dsp to retain the crossover permanently, you can program it into FLASH memory; then the next time the dsp is powered up, it will transfer the crossover from FLASH into SDRAM and automatically execute the crossover for measurements or for playing music.

There are three potentially different crossovers that can exists at any particular time. The dsp can have crossovers in FLASH and/or SDRAM memories, which may or may not be the same; however it will only be able to execute the crossover that is in SDRAM memory, at least until it is re-booted in the next power cycle. Also, the vc.exe program has crossover information, which is referred to as the PC crossover, which may be entirely different from what is in dsp memory. When you start the vc.exe program, it assumes that its PC crossover has not yet been programmed into dsp memory, as is indicated by the final two lines in the Main Menu in Fig. 4. If you subsequently program a crossover into dsp SDRAM or FLASH memory, you will see that information in the Main Menu display, but only during the current vc.exe session.

To summarize, when you power up the dsp, if it detects a crossover already programmed in FLASH memory, it transfers the crossover to SDRAM memory where it is ready for execution. When you start the PC program vc.exe, it looks for a valid crossover identifier in dsp memory and if it finds one, displays it in the Main Menu; however vc.exe does not know any of the details of the crossover in dsp memory at this point. If you want to set up a new crossover on the PC in vc.exe, you can either read previously saved crossover information from a file (option X) or set up the crossover in vc.exe (option L, see below). When you have set up a new crossover that you would like to try, you can program it either in dsp SDRAM memory or FLASH memory (option R). Also, be sure to save any newly developed crossover to a file (option Y).

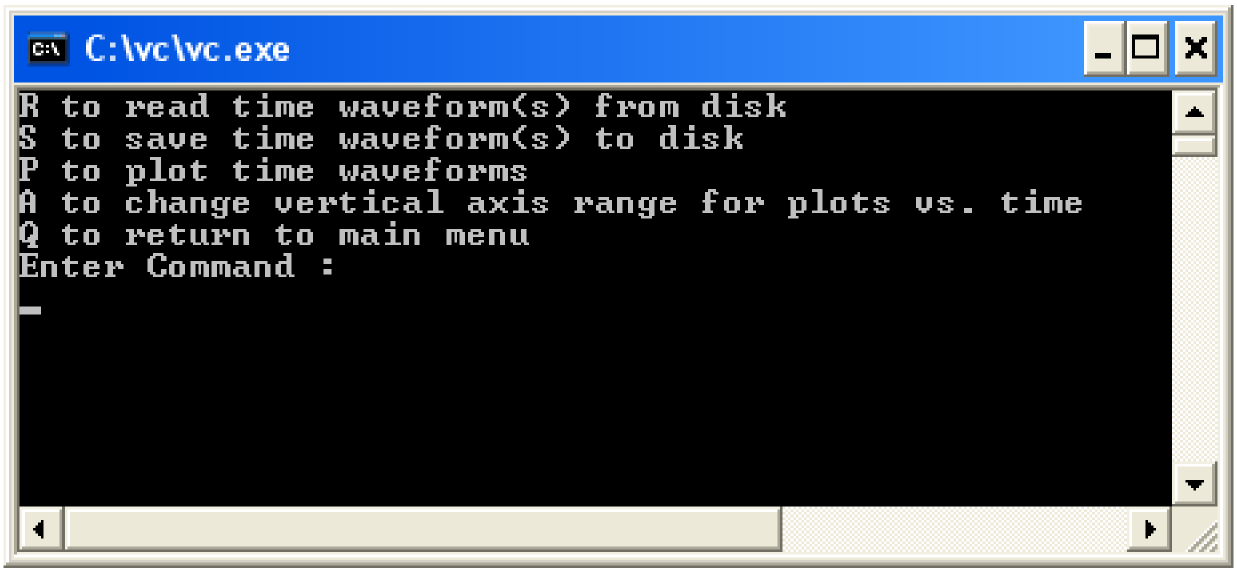
6 P to play music

When the Virtual Crossover is powered up, provided that there is a crossover already programmed in FLASH memory, the device automatically starts in Play Music mode (note: you can change this behavior using the O option from the Main Menu, see below). What this means is that the Virtual Crossover reads stereo input, either from the RCA input jacks or the SPDIF input jack on the back panel, processes the music signals through the crossover filters, and outputs the filtered music signals. However, as soon as the vc.exe program is run from the PC and it starts communicating with the Virtual Crossover, it takes the device out of Play Music mode. This means that when you first see the Main Menu display in Fig. 4, the Virtual Crossover is no longer in Play Music mode. Entering the letter p or P from the PC causes the Virtual Crossover to go into Play Music mode; in this mode the display of the first line in the Main Menu changes, indicating that pressing P again will stop playing music, i.e. entering P again will take the Virtual Crossover out of Play Music mode.

Warning: it is important *not* to enter Play Music mode if the Virtual Crossover is set up to carry out measurements of loudspeakers, i.e. with a microphone output connected to the Virtual Crossover input, and with the Virtual Crossover output going to the speaker. If the Virtual Crossover enters Play Music mode in that situation, it is trying to play the signal that is being output, and destructive levels of feedback can occur which could possibly damage the speaker under test.

7 Time Waveforms Menu

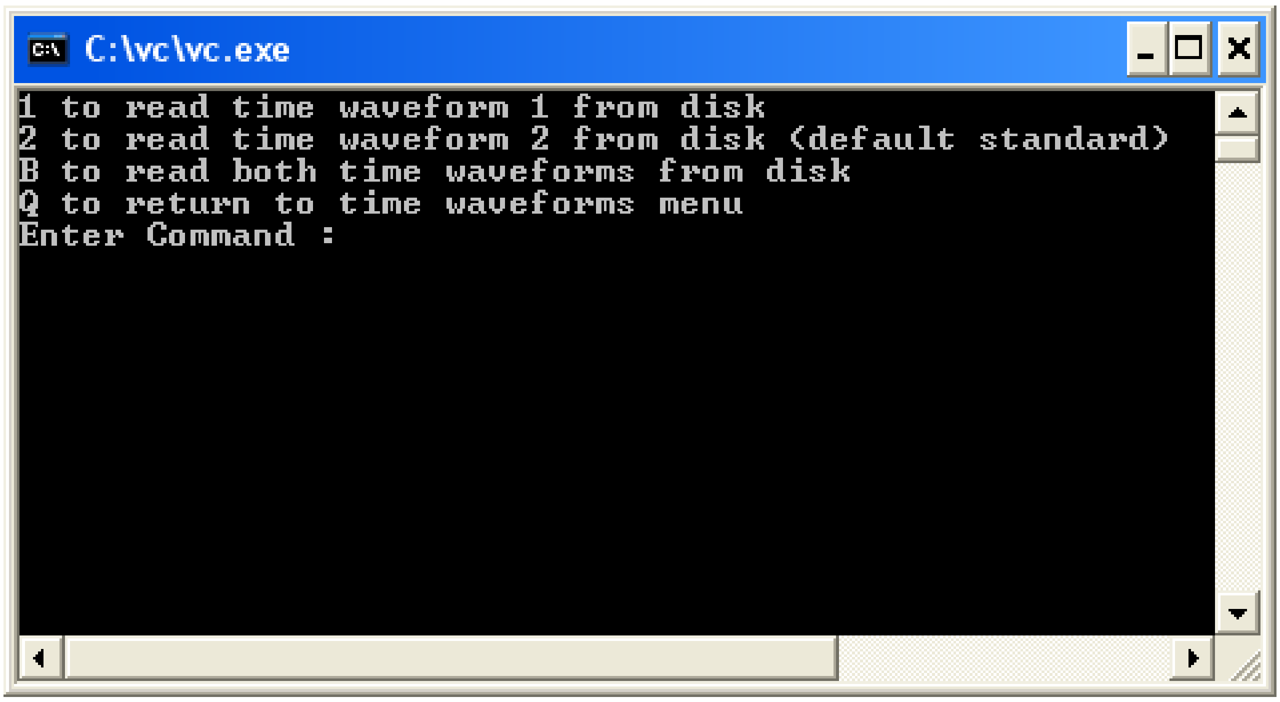
Pressing the letter t or T from the Main Menu takes you to the Time Waveforms Menu display:



```
R to read time waveform(s) from disk
S to save time waveform(s) to disk
P to plot time waveforms
A to change vertical axis range for plots vs. time
Q to return to main menu
Enter Command :
_
```

Figure 5: Virtual Crossover Time Waveforms Menu display.

The Time Waveforms Menu allows you to read and write time-domain waveform files on the PC, and to plot waveform files on the PC screen. If you type R, the following sub-menu appears:



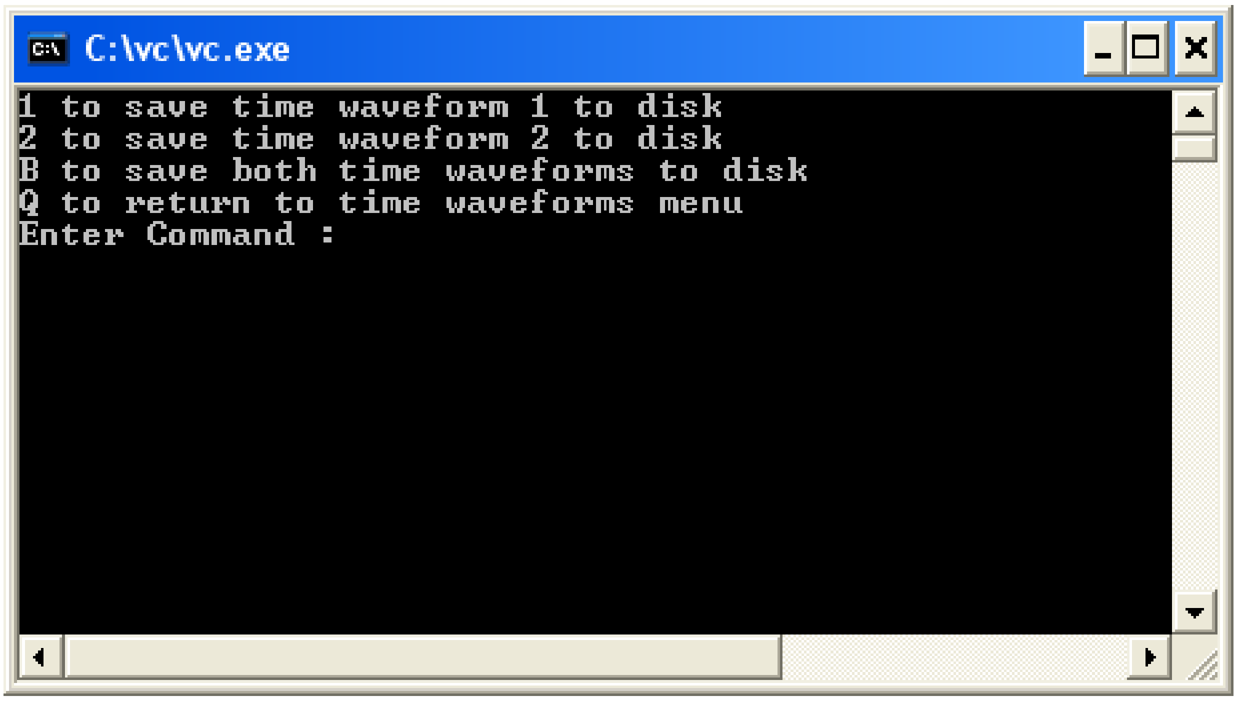
```
C:\vc\vc.exe
1 to read time waveform 1 from disk
2 to read time waveform 2 from disk (default standard)
B to read both time waveforms from disk
Q to return to time waveforms menu
Enter Command :
```

Figure 6: Virtual Crossover Read Time Waveform From Disk display.

The program vc.exe sets up storage for up to two different time-domain waveforms; the above display asks whether you want the file contents to be stored in waveform 1 or 2, or if you want to read two different files into both waveforms. Pressing 1, 2, or B selects one of these options. Note that waveform 2 is referred to as the "default standard"; this is because when the Virtual Crossover is in certain measurement modes, it compares measured data (captured in waveform 1) to a waveform that has been loaded in waveform 2 as the standard. Please see the discussions of the various possible measurement modes later in the manual.

After you have selected 1, 2, or B, the program asks you to enter the filename(s) with the time-domain waveform(s), which must conform to standard format (please see discussion of standard format elsewhere in this manual). vc.exe will complain if it can't find or can't open the file, and it will also complain if it detects a mismatch between the time step used in the file and the time step used in the program (although in this case the program just gives a warning and goes ahead with the load anyway). Warnings are also issued in the following situations: if the number of waveform points in the file is greater than the storage allotted for the waveform, or if the number of waveform points in the file is less than the default number of points available in the program. In the latter case, vc.exe offers the option to fill out the waveform with zeroes beyond the data read in from the file.

Typing S from the Time Waveforms Menu brings up the following sub-menu:



```
1 to save time waveform 1 to disk
2 to save time waveform 2 to disk
B to save both time waveforms to disk
Q to return to time waveforms menu
Enter Command :
```

Figure 7: Virtual Crossover Save Time Waveform To Disk Menu.

From this sub-menu, you can save the waveform either in waveform 1 or 2, or both waveforms to disk. In addition to asking for the filename(s) where to save the waveform(s), vc.exe may also ask whether you want to save the truncated waveform or the entire waveform; this is related to a process that is discussed next with the P or Plot Time Waveforms option.

Typing P from the Time Waveforms Menu in Fig. 5 allows you to plot either waveform 1 or 2 on the PC screen; vc.exe asks you to enter 1 or 2 to identify which waveform you want to plot. In general, there are two ways to acquire time-domain waveforms – one is to read a waveform from disk (see the R option in the Time Domain Waveforms Menu above) or as the result of a measurement (see the Set Up A Measurement Menu below). Let's assume that the latter method has already been used, and that an impulse response measurement of a Vifa D26TG-35 tweeter mounted in a speaker enclosure has been carried out. Measurements are stored in waveform 1 (unless we are in impedance mode, see below), so entering 1 to identify the waveform would bring up the following plot:

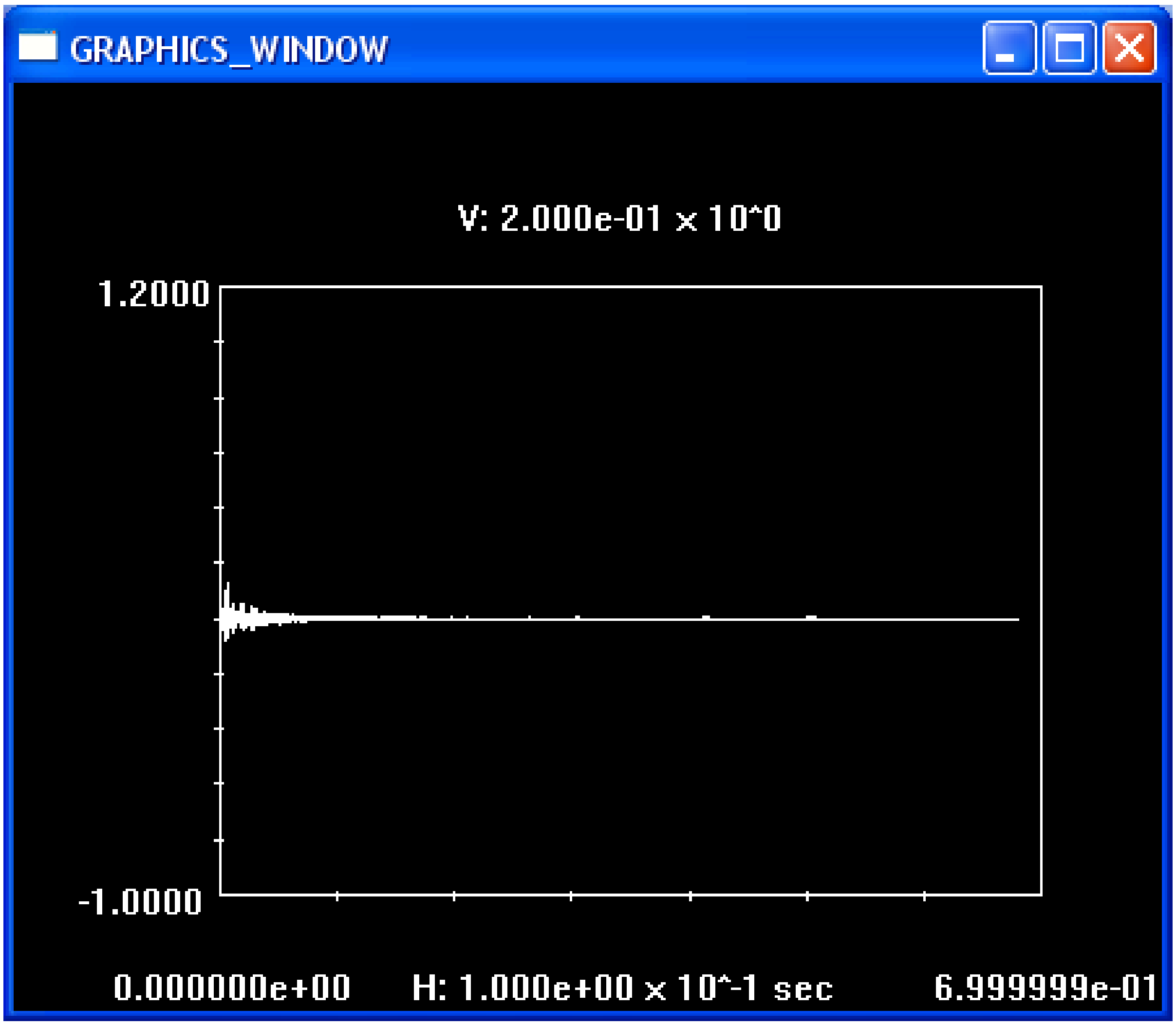


Figure 8: Measured impulse response of Vifa D26TG-35 tweeter.

For this measurement an MLS sequence of order 16 was used (65535 points at 96kHz sampling rate), so the total duration of the measurement is quite long and you cannot see the details of the tweeter primary response. There are various commands you can enter from the keyboard to zoom in on areas of time-domain plots and to truncate the waveforms; here I present a summary of these operations, followed by examples for this particular plot:

C: to enter Cursors mode.
T: Toggles between left and right cursors.
Left or right arrow key: To move selected cursor to the left or right.
A: Accelerates cursor movement each time it is pressed.
B: Decelerates cursor movement each time it is pressed.
D: Returns to default cursor positions.
Q: To exit Cursors mode and select current window.

T: To enter Truncate mode.
Left or right arrow key: To move vertical truncate line to the left or right.
A: Accelerates truncate line movement each time it is pressed.
B: Decelerates truncate line movement each time it is pressed.
D: Returns to default (maximum) truncate line position.
Q: To exit Truncate mode and select current truncation point.

For the first example, let's say that you want to zoom in on the first part of the plot in Fig. 8 to see the details of the early tweeter response. Typing C or c while the plot is active enters Cursors mode. In Cursors mode two vertical cursors appear on the plot, one initially on the left-hand side and the other on the right-hand side. Only one of the cursors is selected at any particular time; initially the left-hand cursor is active, and it is represented by a solid line, while the right-hand cursor is inactive and is represented by a dotted line. Since we want to zoom in on the early portion of the plot, we type T or t to toggle cursors and make the right-hand cursor active. Next hold down the left arrow key to move the right-hand cursor to the left. Since there are so many points in the plot, you will find that the cursor moves very slowly. To speed up the cursor movement, release the left arrow key and press the A or a key (three times in a row is good for this size of plot), which accelerates the cursor movement. Now press and hold the left arrow key again; this time you will find that the cursor moves much more rapidly. After you have moved the cursor over near the left-hand side of the plot, the screen looks something like this:

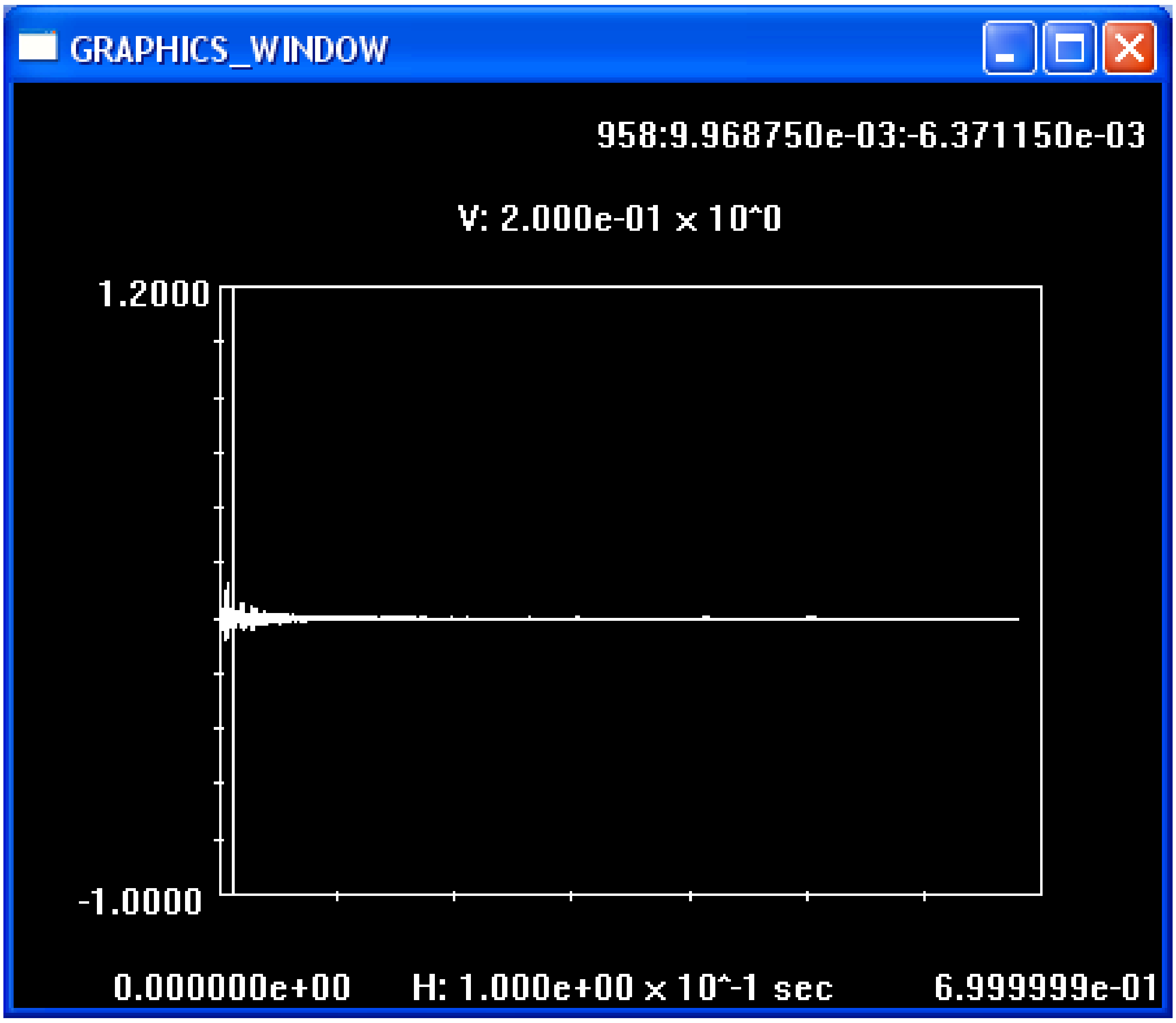


Figure 9: Plot of Fig. 8 in Cursors mode.

Note that as the cursor moves across the screen, a display at the top of the screen keeps track of the following information at each cursor location:

sample number:sample time:amplitude

Pressing D or d at any time causes both cursors to move back to their original positions, at the extreme left- and right-hand extents of the data.

If you press Q or q while the right-hand cursor is in the position shown in Fig. 9, Cursors mode is exited and the plot is transformed as follows:

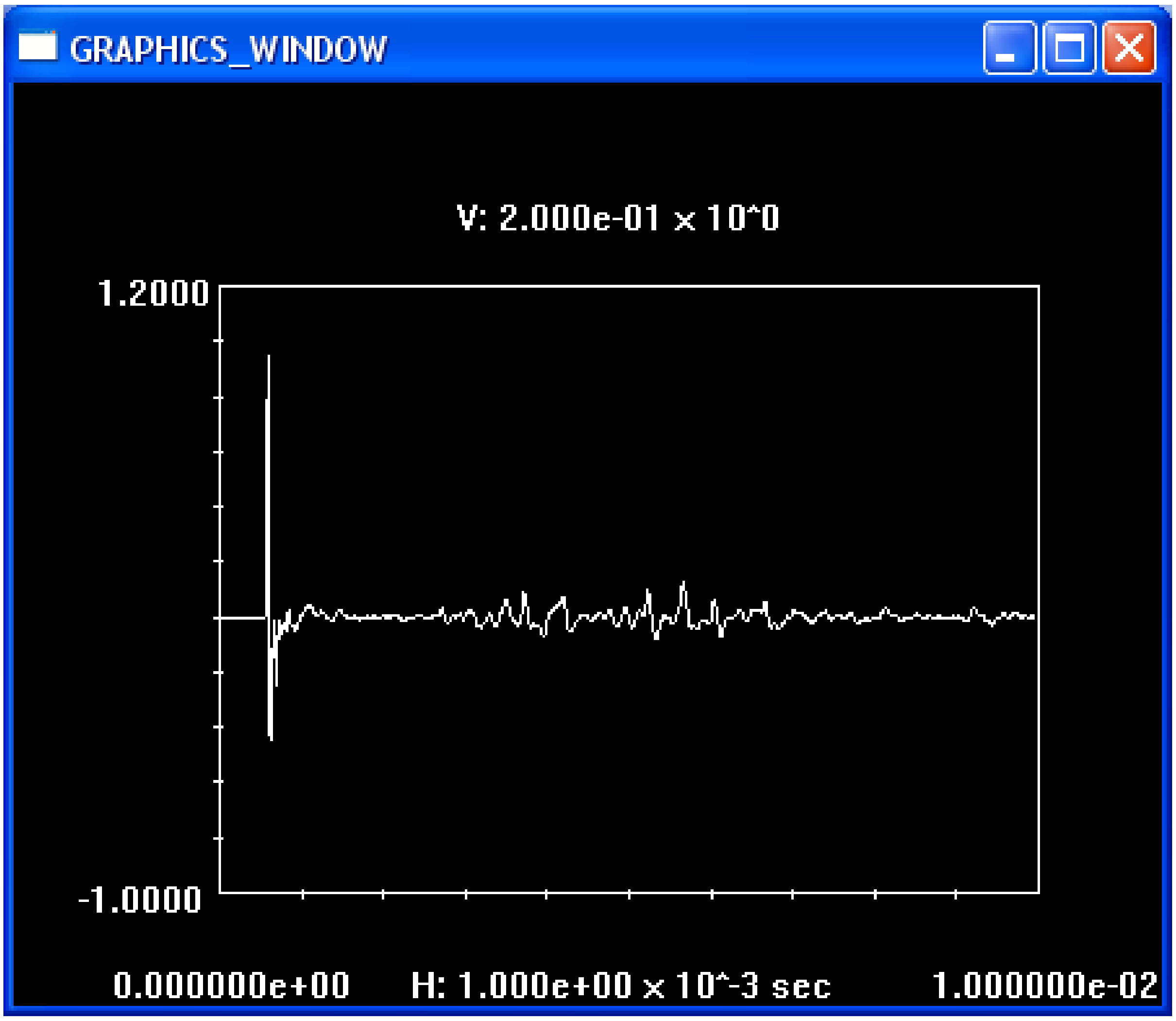


Figure 10: Plot re-scaled to include only the time interval indicated in Fig. 9.

where now only the time interval between the cursors in Fig. 9 is displayed.

It can be seen in Fig. 10 that the primary tweeter response occurs in about the first 2 milliseconds, whereas afterwards a rather long transient consisting mostly of room reflections occurs. To do a "quasi-anechoic" measurement, i.e. to do a measurement approximating what we would get in an anechoic chamber, we need to truncate the waveform in Fig. 10 at some point in time after the primary response of the tweeter, but before the room reflections enter the picture. Typing T or t enters Truncation mode, where a vertical line appears initially on the right-hand side of the plot. Truncation mode is somewhat similar to Cursors mode, in that you can move the vertical line with the left and right arrow keys, and the movement can be speeded up or slowed down by typing the A or B keys, respectively; however in Truncation mode only a single vertical line can be moved. Also similar to Cursors mode, typing D or d returns the vertical truncation line to its right-most default position, and a display at the top of the plot keeps track of the truncation line statistics. Let's say that we position the truncation line at a point that seems to be a good demarcation between the primary response and the onset of room reflections, as follows:

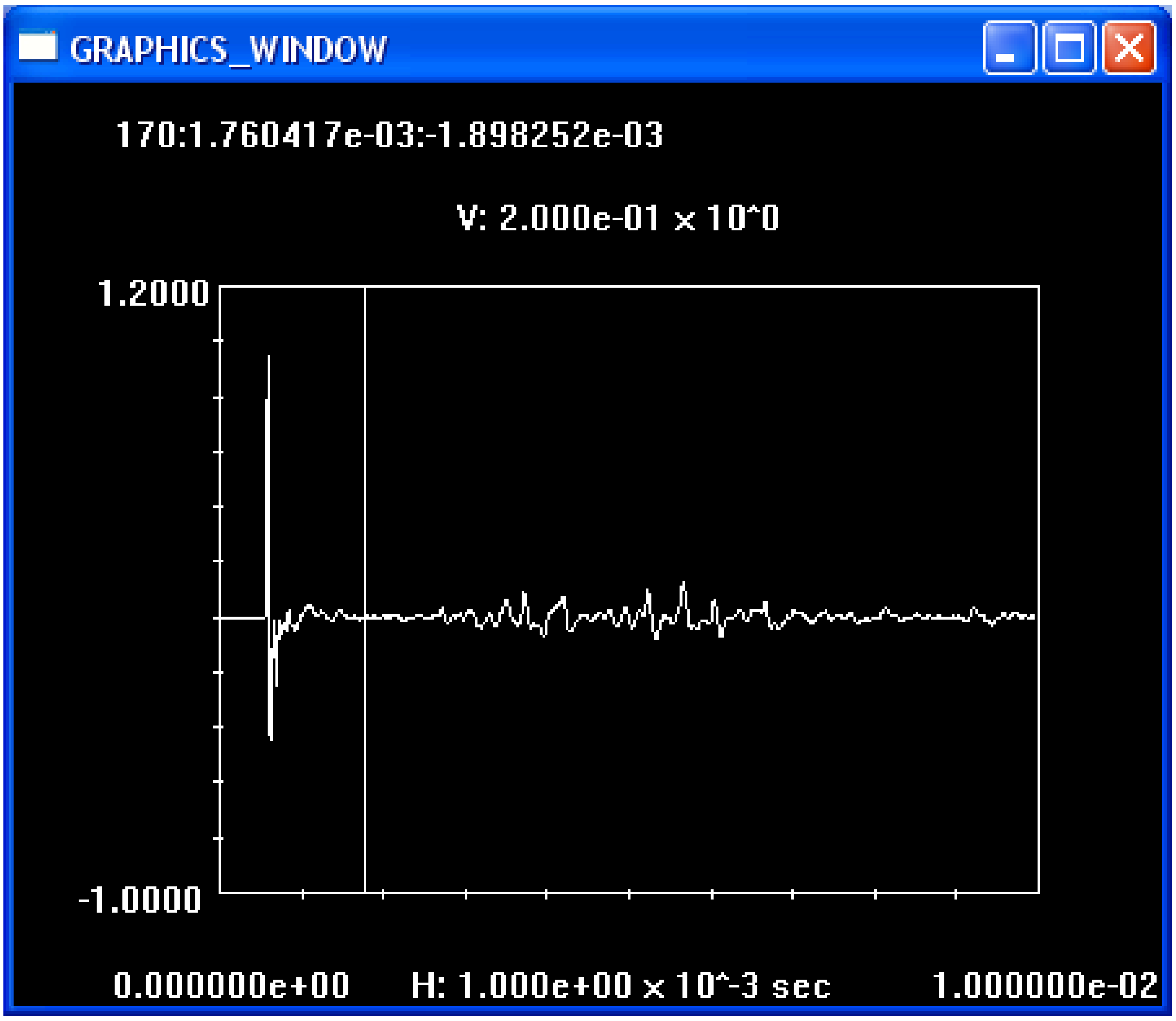


Figure 11: Plot of Fig. 10 in Truncation mode with vertical line at one possible truncation point between primary tweeter response and room reflections.

Pressing Q or q at this point exits Truncation mode, and transforms the plot as follows:

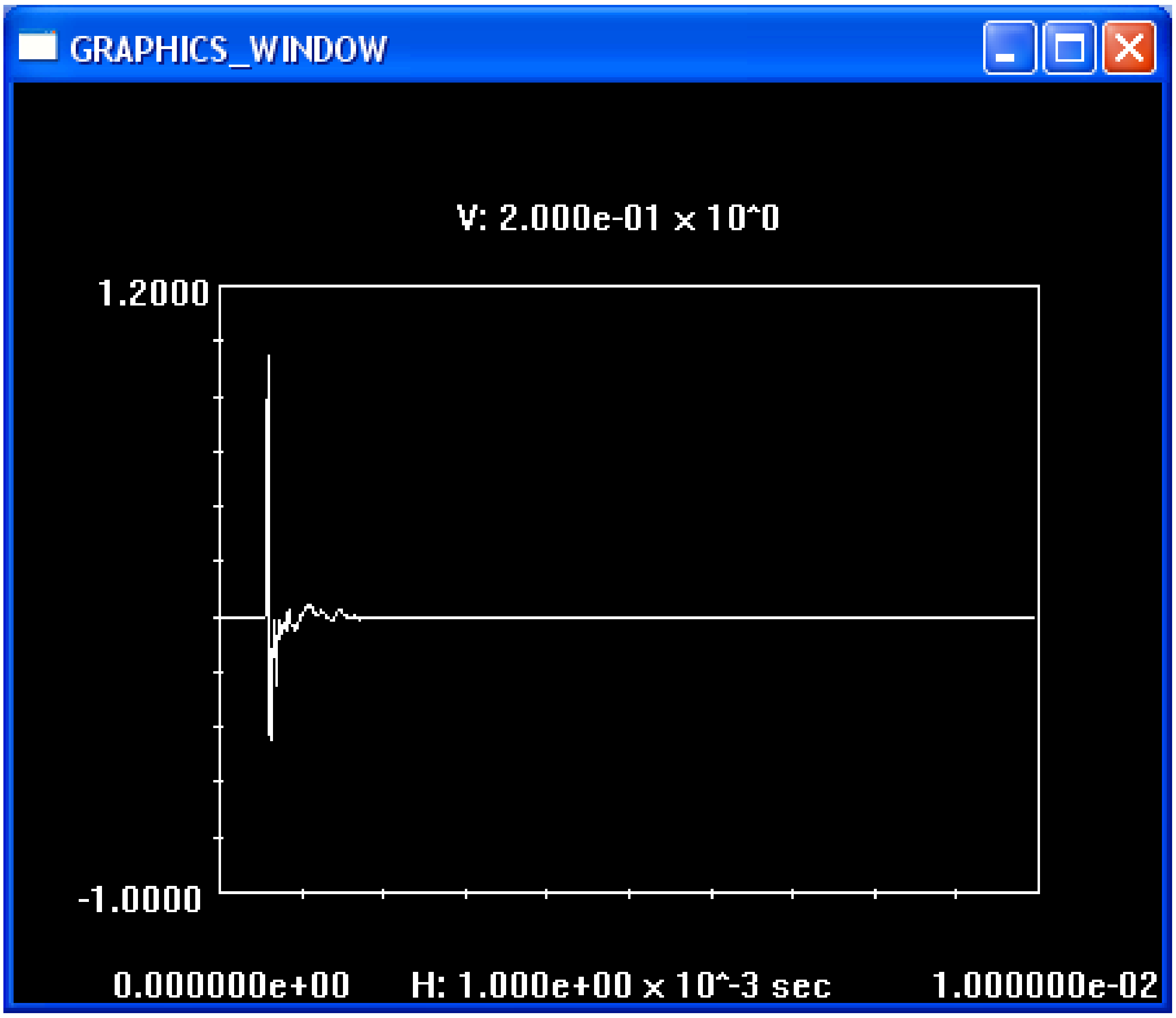


Figure 12: Plot from Fig. 10 after truncation at point number 170.

The initial portion of the plot in Fig. 12 is the same as in Fig. 10, however beyond point number 170 at 1.76ms the waveform is now assumed to be zero. Truncation is important when frequency-domain results are calculated from the time-domain waveforms; please see the explanation of the FFT Menu below for examples. Once truncation points have been set for either waveforms 1 or 2, they are used in subsequent frequency-domain calculations.

Finally, the A option from the Time Waveforms Menu can be used to change the default vertical axis settings for time-domain plots. For example, the plot in Fig. 12 would be better displayed if the vertical axis limits were about +1 and -0.6. Typing A or a from the Time Waveforms Menu causes vc.exe to ask that you enter new vertical axis limits. If we type A and enter the limits 0.9 and -0.6, and then plot waveform 1 again, we get the following display:

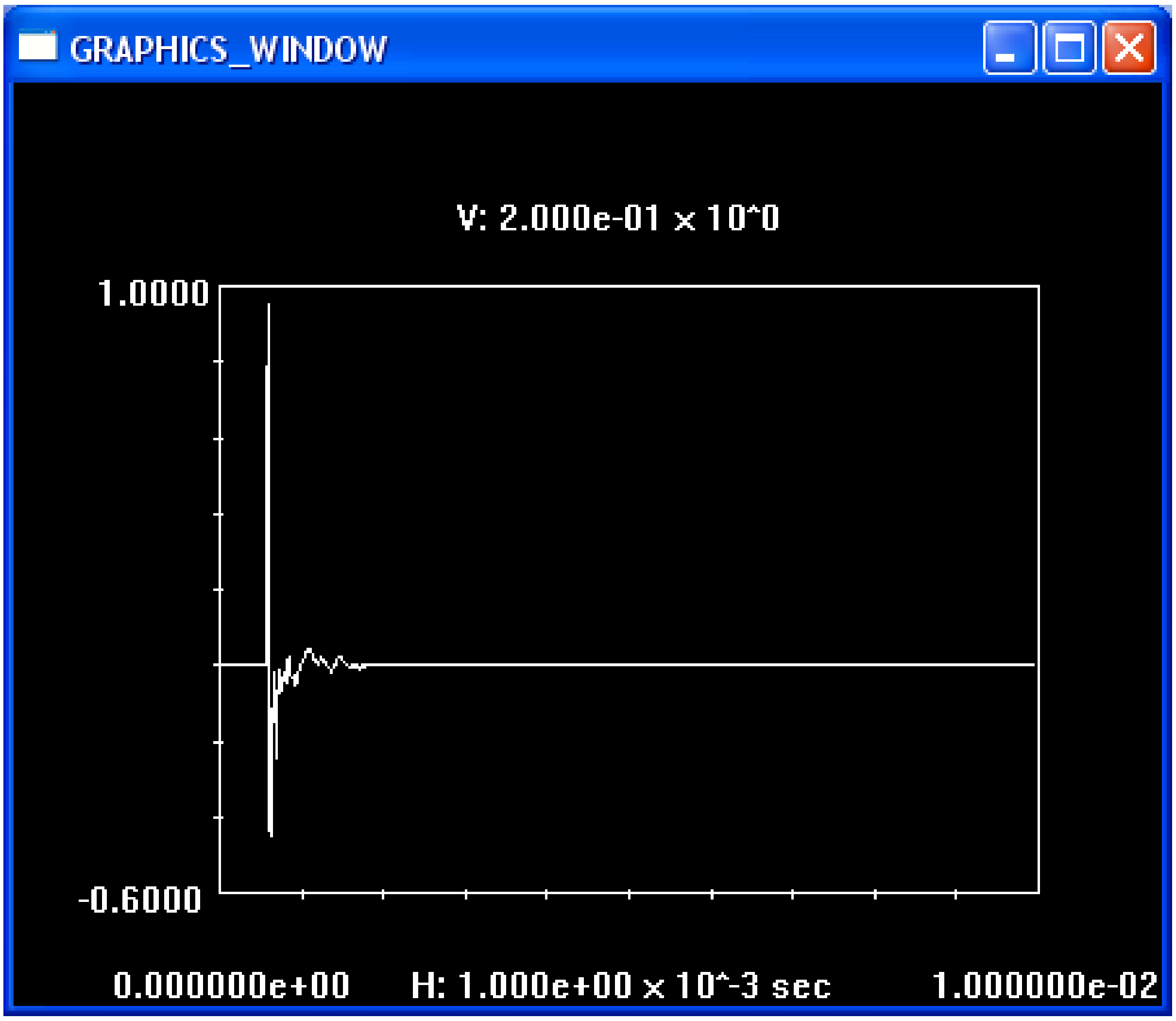
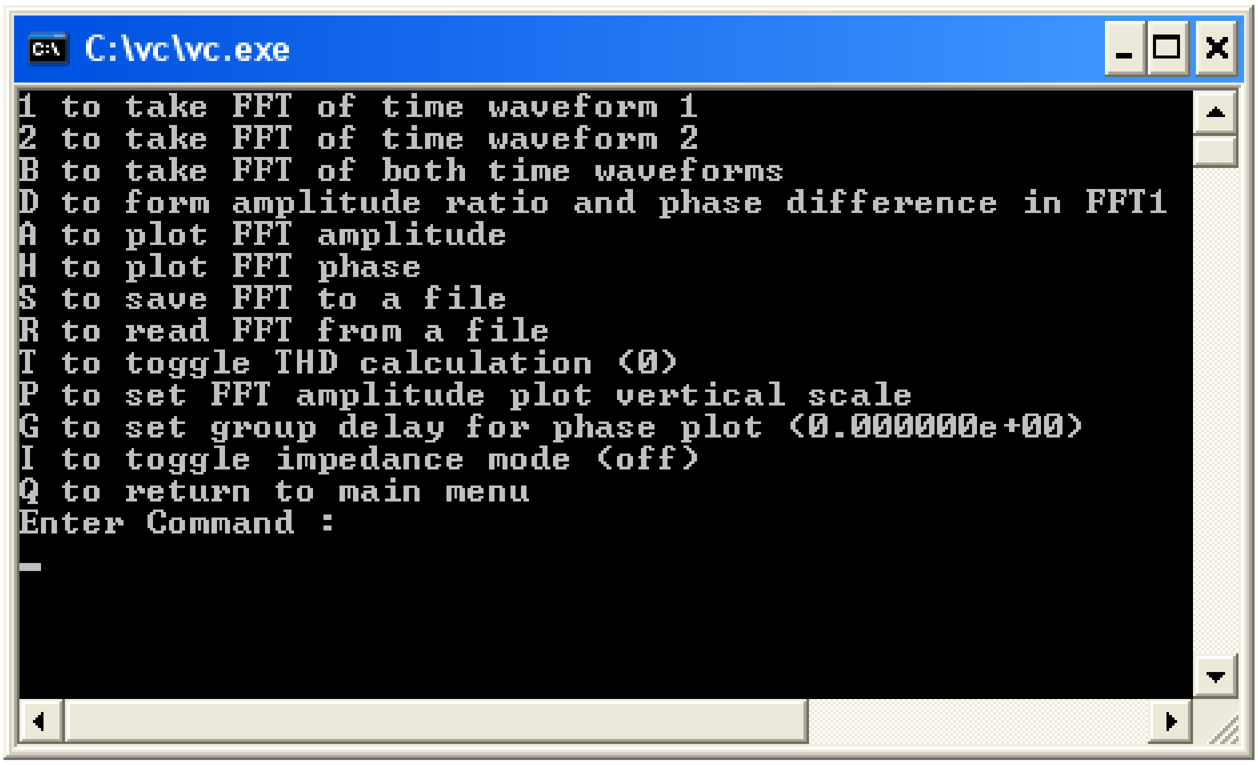


Figure 13: Plot of Fig. 12 after adjusting vertical axis limits using the A option.

Note that in Fig. 13 we didn't end up with exactly the axis limits that we requested. The vc.exe program uses its own algorithm to decide on the final axis limits, which may be somewhat different than the ones requested. So a little experimentation may be necessary to get exactly the axis limits that you want.

8 FFT Menu

Pressing F or f from the Main Menu brings up the FFT Menu display:



```
1 to take FFT of time waveform 1
2 to take FFT of time waveform 2
B to take FFT of both time waveforms
D to form amplitude ratio and phase difference in FFT1
A to plot FFT amplitude
H to plot FFT phase
S to save FFT to a file
R to read FFT from a file
T to toggle THD calculation (0)
P to set FFT amplitude plot vertical scale
G to set group delay for phase plot (0.000000e+00)
I to toggle impedance mode (off)
Q to return to main menu
Enter Command :
_
```

Figure 14: FFT Menu display obtained by typing F from the Main Menu.

Since the FFT (Fast Fourier Transform) transforms from the time domain to the frequency domain, you must have at least one time-domain waveform in place before you can carry out an FFT. Pressing 1 will take the FFT of the time-domain waveform 1, and 2 will take the FFT of time-domain waveform 2; pressing B takes the FFT of both waveforms. Let's assume that we still have the truncated impulse response of the Vifa D26TG-35 tweeter from Fig. 12 in waveform 1, and we press 1 while in the FFT Menu (just pressing 1 is sufficient, you don't need to press enter afterwards). We can then plot the results for the magnitude of the FFT by first typing A from the FFT Menu and then entering 1 when asked which FFT to plot. Here is the result for the impulse response waveform of Fig. 12:

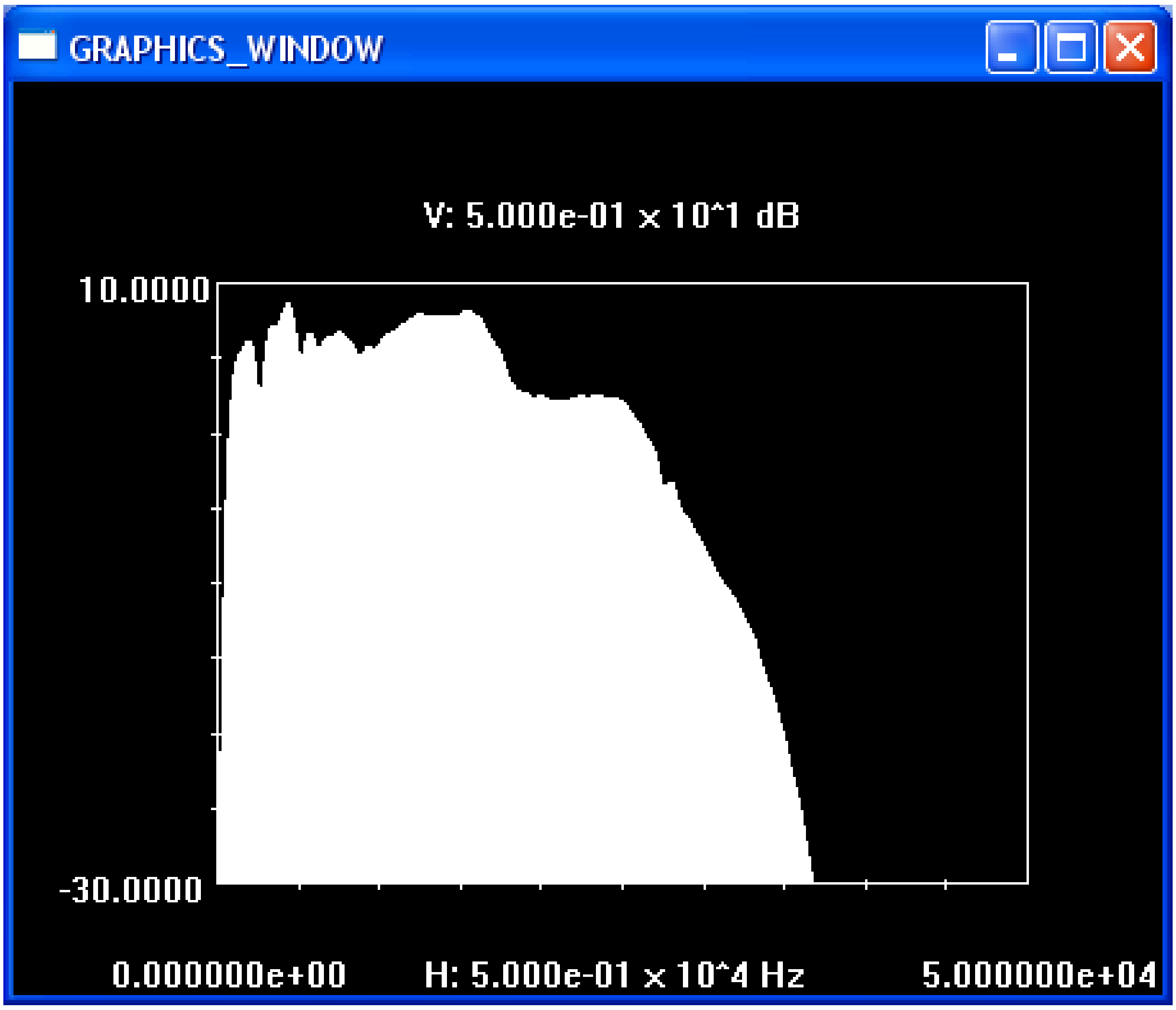


Figure 15: Magnitude of the frequency response of the waveform in Fig. 12.

First some comments about the ranges in Fig. 15 that may be helpful in interpreting these plots. Since the sampling rate for Fig. 12 is 96kHz, the maximum frequency (or folding frequency, if you are familiar with that term from digital signal processing theory) is 48kHz or half the sampling frequency. This is the maximum frequency for which data is included in Fig. 15, although the tweeter response drops below -30dB before we reach 48kHz so part of the response is not visible on the graph; adjusting the lower vertical axis limit (option P) would allow us to see the tweeter response in this region. Also, since there are a total of 65536 points in Fig. 12, the total time for the waveform is $65536/96000 = 0.6827$sec, which implies that the frequency resolution (the distance between vertical lines in Fig. 15, which you can't really see) is $1/0.6827 = 1.465$Hz.

Although it is not specifically indicated in the FFT Menu, typing C brings up a Cursors Mode which is identical to the Cursors Mode for time-domain waveforms; please see the discussion of the Cursors Mode in the Time Waveforms Menu section. Here, for example, we might want to focus attention on the tweeter response only out to 20kHz; typing C and then T (toggle) to activate the right-hand cursor, followed by pressing the left arrow key (the A and B keys speed up and slow down cursor movement, just as for time-domain waveforms) until the right-hand cursor is just below 20kHz, and finally pressing Q to exit Cursors Mode and to accept the new window, we end up with the following plot:

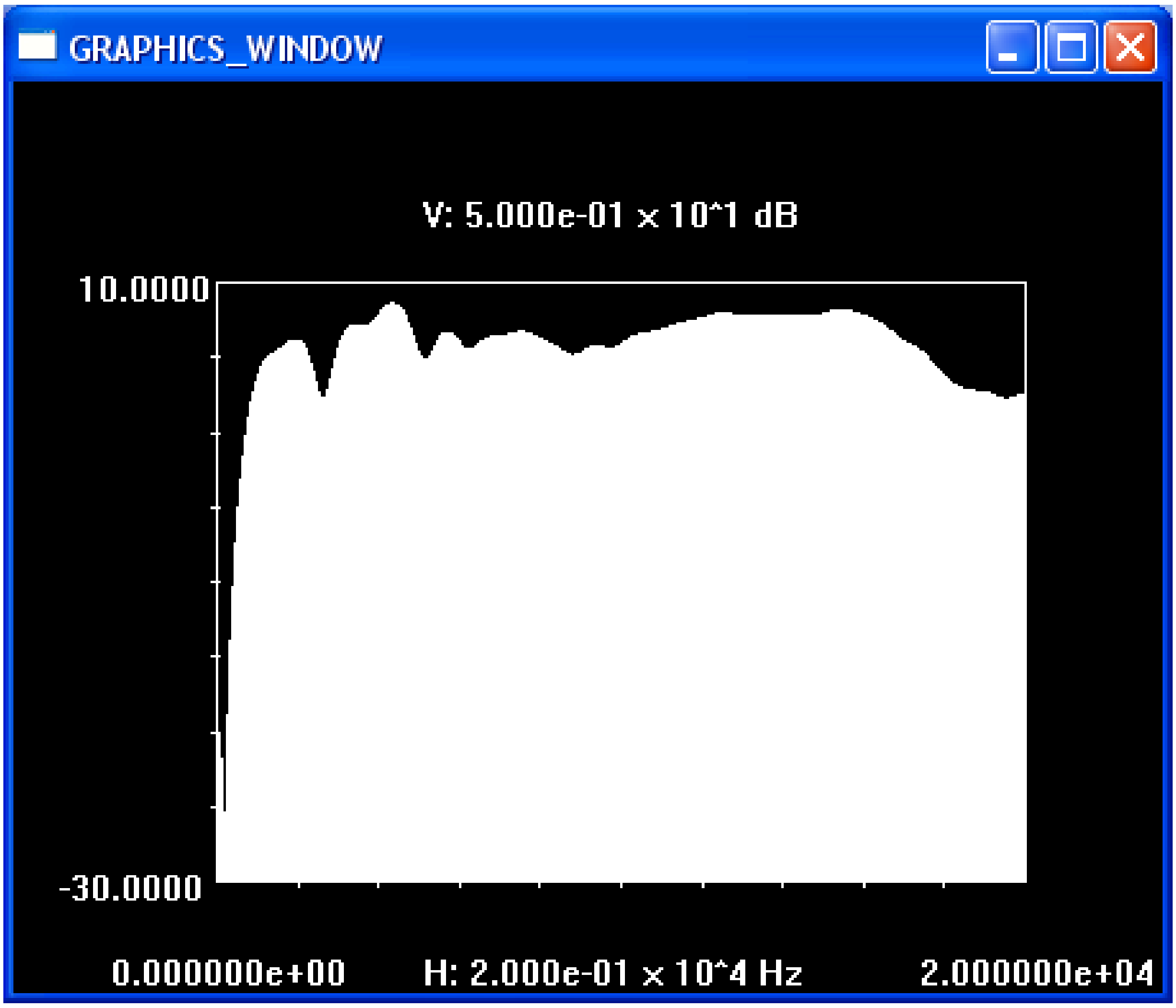


Figure 16: Frequency response of Fig. 12 restricted to the 0-20kHz range, using Cursors Mode.

Another thing we might want to do is adjust the maximum and minimum limits on the vertical axis. It was mentioned previously that we could make the lower limit more negative to see the tweeter response at lower levels. For now, however, let's just adjust the levels to better center the plot in the vertical direction. If we type P, the program asks for the new maximum and minimum limits for the vertical axis in dB; if we enter, for example, 20 and -20, the following graph results:

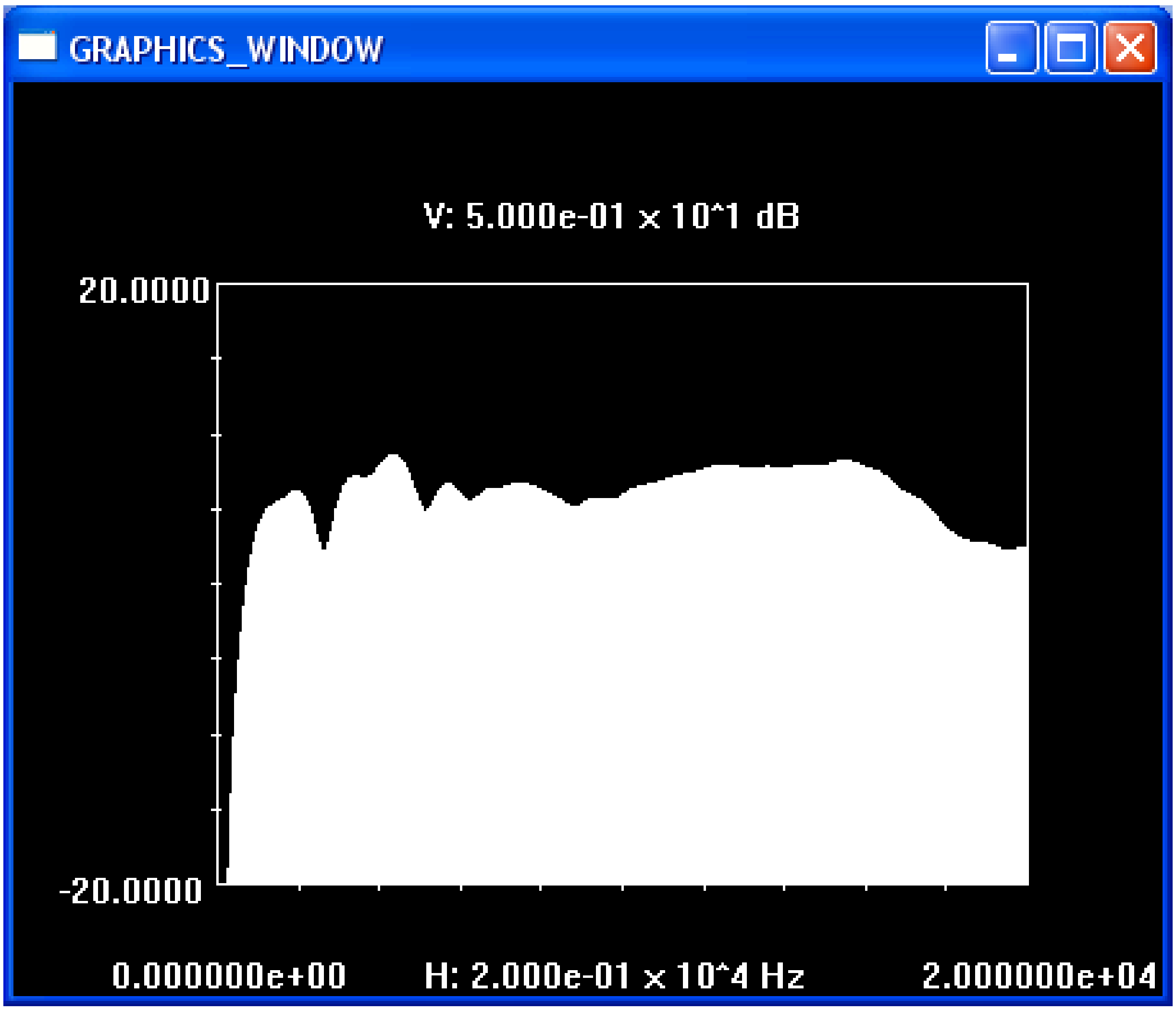


Figure 17: Frequency response of Fig. 12 with adjusted vertical axis limits, using the P option.

(It should be mentioned that the frequency response in Fig. 17 not only shows the Vifa tweeter response, it also includes the effects of the speaker enclosure as well as reflections from the other drivers in the system).

In general, the frequency-domain response of a time-domain waveform is a complex quantity, i.e. it contains both a real and imaginary part; instead of displaying real and imaginary parts, vc.exe displays magnitude and phase. We already saw the magnitude response in Fig. 17. These quantities are related as follows. First of all, the magnitude M is related to the magnitude in dB, M_{dB}, by:

$$M_{\text{dB}} = 20\log_{10}(M) \tag{1}$$

Let the real part be denoted by Re and the imaginary part by Im, and let θ denote the phase angle in degrees. Then:

$$Re = M\cos(2\pi\theta), \tag{2}$$

$$Im = M\sin(2\pi\theta). \tag{3}$$

The H option in the FFT Menu is for plotting phase response. The program asks you to enter either 1 or 2 to identify which phase response you want to plot. If we press H and then enter 1, the following phase plot results:

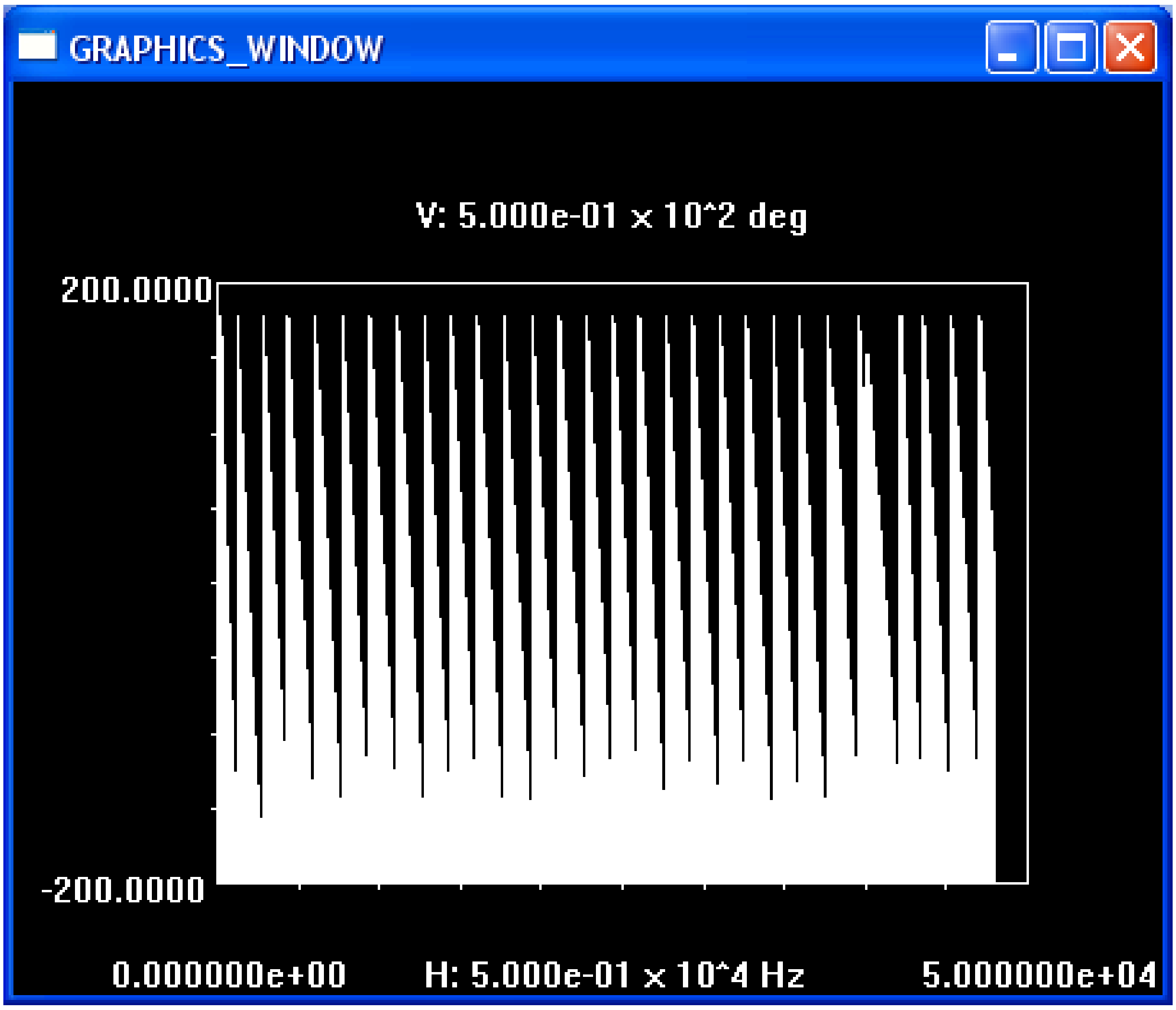


Figure 18: Phase response of Fig. 12 obtained using the H option.

Note that the phase response is by default plotted over the same frequency range as for the magnitude response in Fig 15. The phase response is plotted in degrees, and it always stays inside the range $-180°$ to $180°$. The phase response can be tricky to interpret; it is very much dependent on the time delay of the waveform being analyzed. From the time-domain plot in Fig. 12, it appears that the waveform starts around 0.6ms, in other words that the waveform is delayed by that amount. From digital signal processing theory, delaying a waveform by an amount τ seconds in time introduces a phase term $e^{-j2\pi f\tau}$ in the frequency domain, or a negative linear phase term of slope $2\pi\tau$. The rapidly varying phase response in Fig. 18 in the negative direction is a manifestation of this time delay. Unfortunately this kind of rapid variation masks the details of the phase response that we would like to see, so the vc.exe program offers an option to compensate for any delay in the time-domain waveform. The G option from the FFT Menu allows the user to specify a group delay to subtract from the phase response. For example, if we type G and then enter 0.6e-3 to subtract out 0.6ms of delay, then typing H and entering 1 again for a new phase plot results in the following:

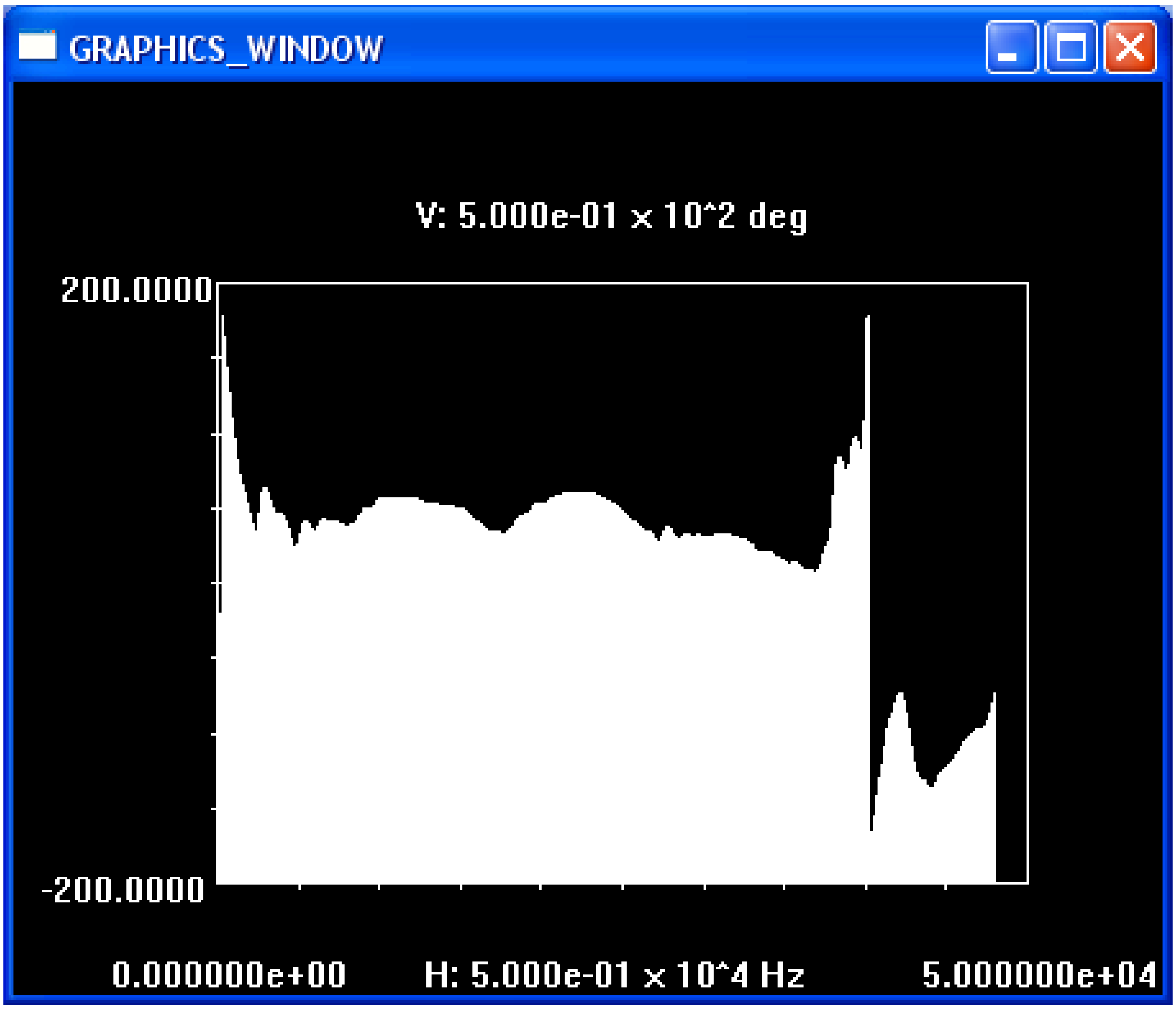


Figure 19: Phase response in Fig. 18 after removing 0.6ms delay using the G option.

An ideal phase response would be linear, so you can see from Fig. 19 that there is phase distortion in the tweeter response (as there is with all drivers that I have ever measured).

We can use the Cursors Mode just as we did for time-domain waveforms and for the magnitude of the frequency response, to for example zero in on the range 0-20kHz for the phase response. Typing C, then T (toggle) to activate the right-hand cursor, then pressing the left arrow key (A and B can be pressed as before to speed up or slow down cursor movement) until the right-hand cursor is just below 20kHz results in the following phase response plot:

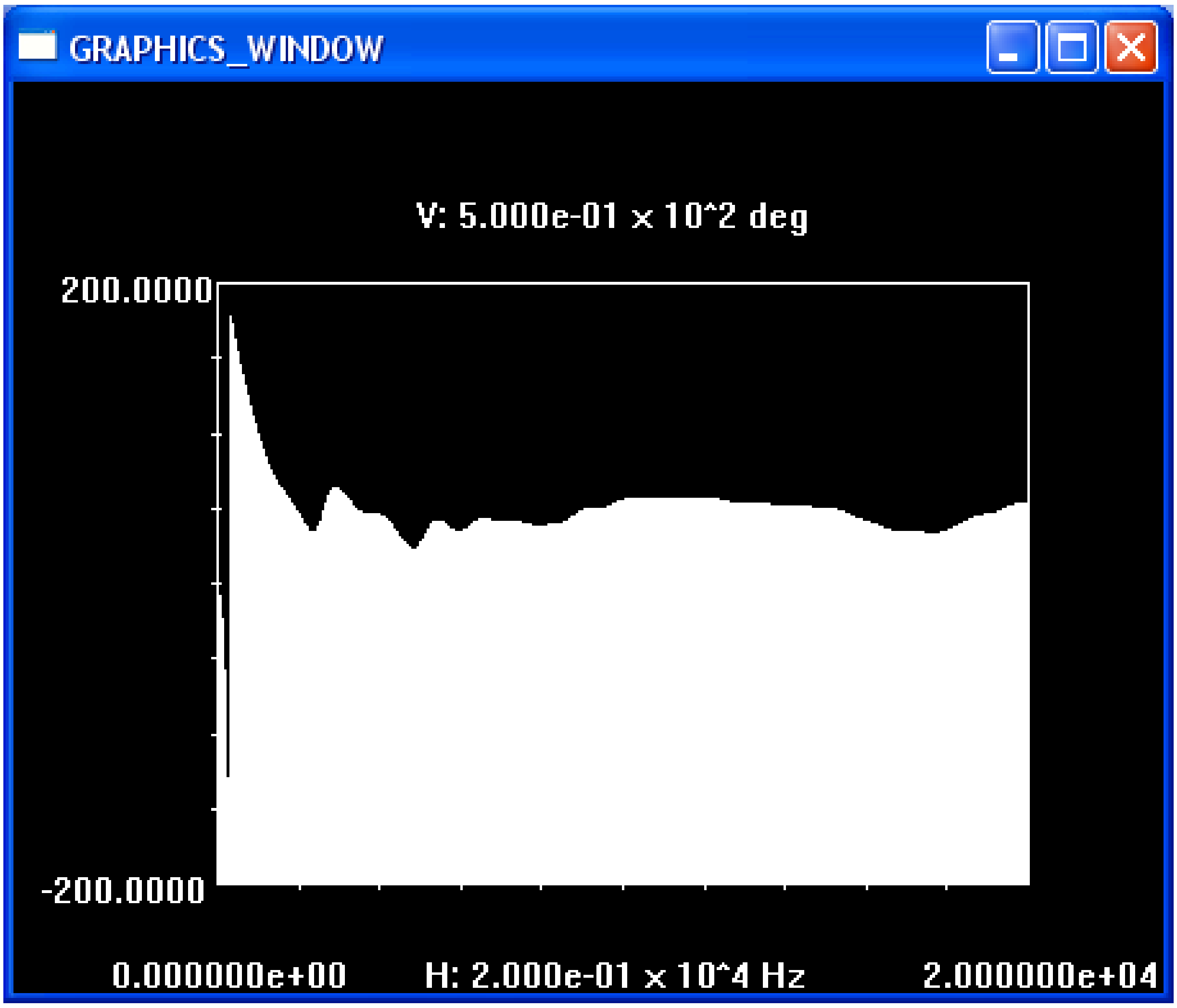


Figure 20: Phase response of Fig. 19 after using Cursors Mode to focus attention on the 0-20kHz range.

The D option in the FFT Menu is used to take the ratio of the frequency-domain response of waveform 1 to that of waveform 2. Let $Re1(\omega) + jIm1(\omega)$ denote the complex values of the frequency response of waveform 1 as a function of $\omega = 2\pi f$, and $Re2(\omega)+jIm2(\omega)$ denote the same values for waveform 2. Then typing D or d from the FFT Menu does the following:

$$\frac{Re1(\omega) + jIm1(\omega)}{Re2(\omega) + Im2(\omega)} \rightarrow Re1(\omega) + Im1(\omega), \tag{4}$$

in other words the complex ratio of the frequency response of waveform 1 over that of waveform 2 is computed and placed back in the frequency response of waveform 1. (Note that before the D option can be selected, the FFTs of both waveforms 1 and 2 must be calculated first). There are normally two situations where the D option is used. The first is where we need to reference a measurement to a standard. For example, let's say that a pulse of some kind is used to test a loudspeaker, and that the frequency response of the pulse is not flat; in that case frequency response 2 would be that of the pulse, and we would compare the measurement result captured in frequency response 1 to it. That way, if the response of the speaker is the same as the pulse, the result of the measurement would be flat, i.e. constant magnitude and linear (or zero) phase shift. Another similar application might be if we know that the amplifier driving the speaker under test alters the input signal in some well-defined manner, then we could put this function into waveform 2 to compensate for it.

The other major use for the D option is when we are in impedance mode. In that case, waveform 1 represents the voltage across the device under test, and waveform 2 is the current into the device. In order to obtain the input impedance, we must take the ratio of these two quantities.

Taking the complex ratio of the two waveforms is equivalent to subtracting the magnitudes (when expressed in dB) and subtracting the phase angles. The D option is implemented immediately when D is pressed, you do not need to press the Enter key. Also, on a fast computer the option may be completed so fast that there is hardly any visual indication that the calculation has been completed.

The R and S options in the FFT Menu are similar to the corresponding options to read or save files from the Time Waveforms Menu. The difference here is that it is assumed all files are in frequency-domain format. For data lines, this format contains 3 numbers per line: the first is the frequency, the second the magnitude (in dB if not in impedance mode, in Ohms if in impedance mode), and the third is the phase angle in degrees.

The I option in Fig. 14 toggles Impedance Mode. As its name implies, you switch on this option when you want to measure the input impedance of a network. When you press I, the program asks for the value of series resistance that you will use for the impedance measurement. To carry out an impedance measurement, you must connect a resistor in series with the network or device being tested. Generally the value of the resistance should be on the same order as the expected impedance magnitude of the device under test, and it is important to accurately know its value.

Since typically in speaker crossover or driver testing we are dealing with fairly low impedances, it is usually necessary to drive the device being measured with a power amplifier rather than directly with the output of the Virtual Crossover. A typical configuration for impedance measurement is to connect the Virtual Crossover 1L output to the power amplifier input. To capture the impedance data, the point at the power amplifier output (on the side of the series resistor that is connected to the power amplifier) is connected to the Input-L input of the Virtual Crossover, and the side of the series resistor connected to the device under test is connected to the Input-R input of the Virtual Crossover. In this configuration, the difference between the signals at Input-L and Input-R gives the current into the device (after dividing by the series resistance), and the signal at Input-R gives the voltage across the device; the ratio of these two quantities yields the device input impedance.

The above assumes that the power amplifier output can be grounded at the same ground reference as the Virtual Crossover. If there is a problem grounding the output of the power amplifier, an alternate configuration can be used whereby each of the above voltages is measured in separate steps. For example in the first step the voltage across the series resistor would be measured by Input-L and Input-R, and in the second step the input voltage across the device would be measured by Input-L at one device input and Input-R at the power amplifier ground reference.

For impedance measurement, the Virtual Crossover is usually configured so that the 1L output channel (the channel used to drive the device under test through the power amplifier) does not have a filter associated with it, i.e. it has a negative filter index assigned to it, as explained in the section on the L option for setting up crossover filters.

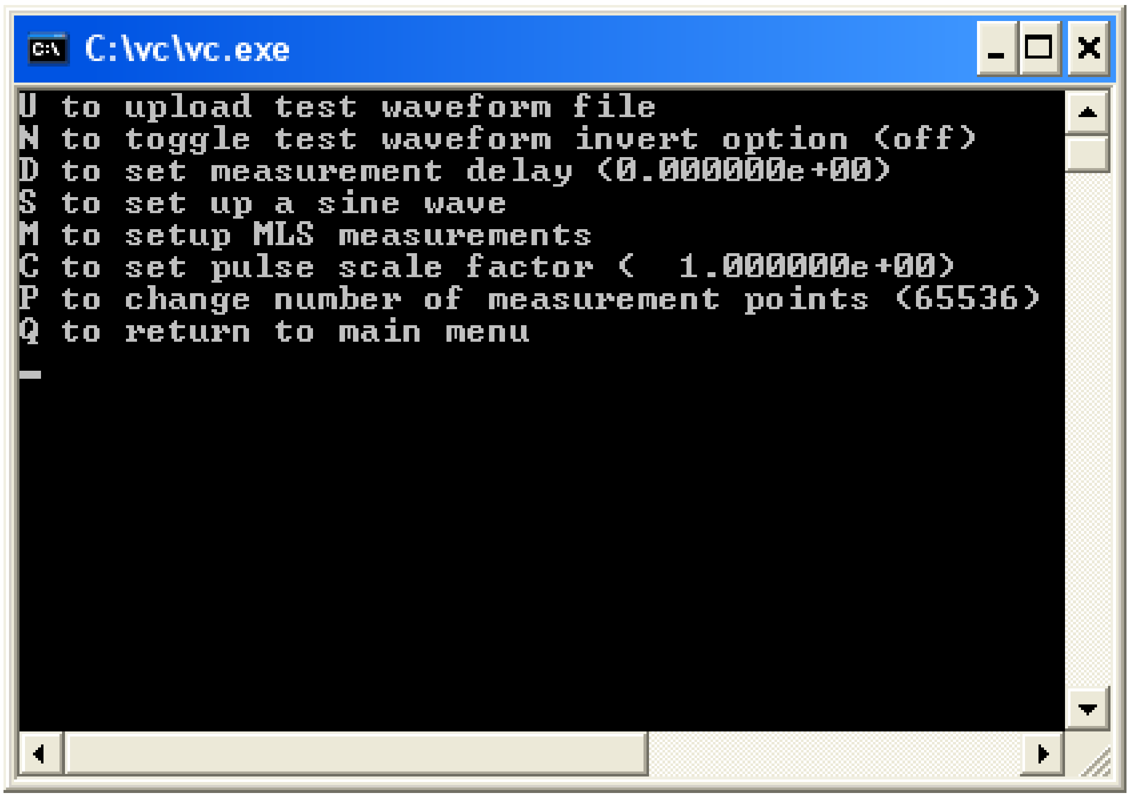
Selecting impedance mode affects the way in which the vc.exe program manipulates measurement data. For example, when impedance mode is selected, calculated FFT magnitudes are not converted to dB, since we need to take the ratio of unconverted magnitudes to obtain impedance values in Ohms. Impedance mode also affects how the D option in the FFT Menu is carried out; normally the D option computes the ratio of the FFT magnitudes, but when impedance mode is selected it must first take the difference of the two waveforms and divide by the series resistance to obtain the current, and then take the ratio of voltage and current. Also, FFT magnitude plots (A option in the FFT Menu) are shown in units of Ohms instead of dB.

Note: bring up the amplitude slowly while monitoring waveforms on the PC screen in impedance mode, in order to avoid applying excessive amplitudes to the Virtual Crossover inputs.

Please see the example of input impedance measurement in the Measurement Examples section for further information on using Impedance Mode.

9 M To Setup A Measurement from Main Menu

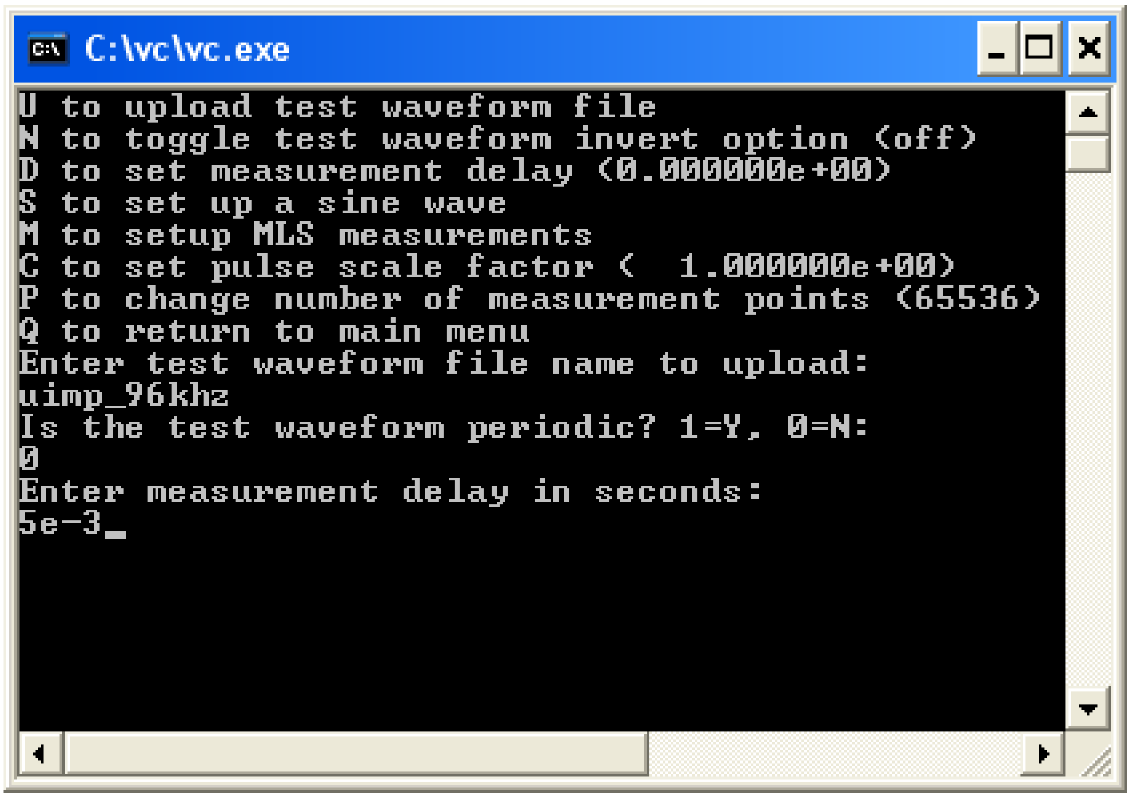
Typing M or m from the Main Menu brings up the Setup A Measurement Menu:



```
U to upload test waveform file
N to toggle test waveform invert option (off)
D to set measurement delay (0.000000e+00)
S to set up a sine wave
M to setup MLS measurements
C to set pulse scale factor (  1.000000e+00)
P to change number of measurement points (65536)
Q to return to main menu
_
```

Figure 21: Set Up A Measurement Menu (option M from Main Menu).

There are three different mechanisms for setting up measurements built into the vc.exe program. First is the ability to load an arbitrary waveform read in from a file; this is the U option in Fig. 21. As an example, I will load a file called uimp_96khz, which is just a unit impulse with sampling rate of 96kHz. Typing U and filling in the details for this file upload results in the following screen:

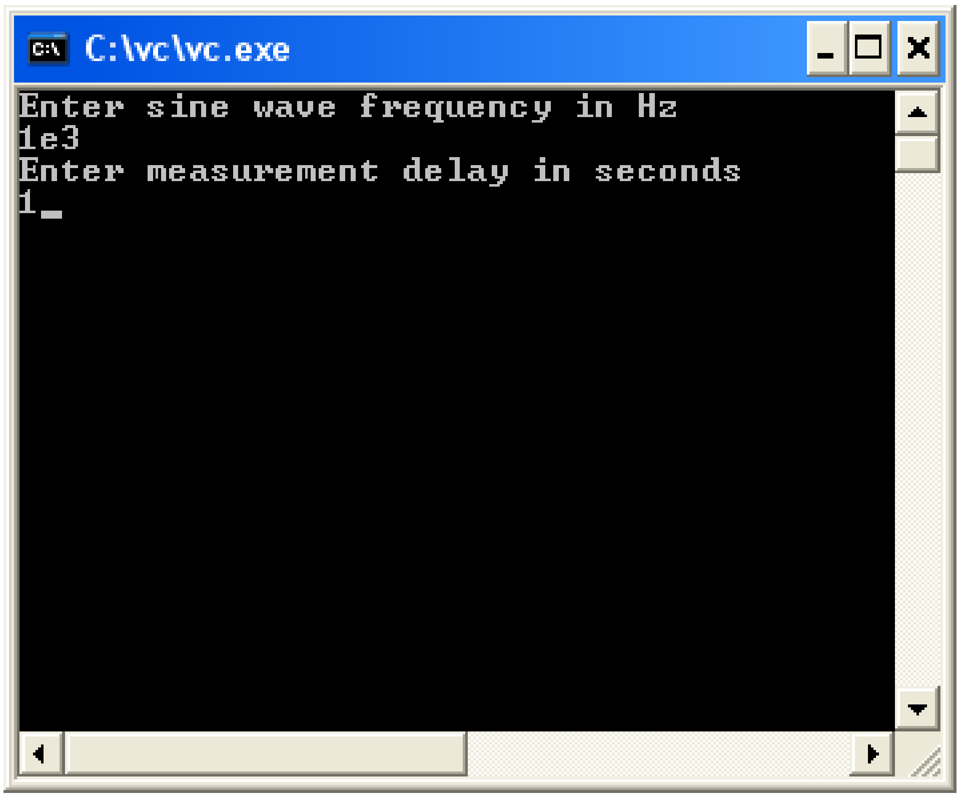


```
U to upload test waveform file
N to toggle test waveform invert option (off)
D to set measurement delay (0.000000e+00)
S to set up a sine wave
M to setup MLS measurements
C to set pulse scale factor (  1.000000e+00)
P to change number of measurement points (65536)
Q to return to main menu
Enter test waveform file name to upload:
uimp_96khz
Is the test waveform periodic? 1=Y, 0=N:
0
Enter measurement delay in seconds:
5e-3_
```

Figure 22: Uploading a measurement waveform called uimp_96khz.

The program first asks for the name of the file (which must be in standard format). It then asks if the waveform is to be periodic. If you type 1 (yes) for this option, it is assumed that the waveform in the file is just one cycle of a waveform that will be repeated without any gaps between cycles. This option would be appropriate for a waveform such as a sine wave, for example. In this case the measurement waveform is meant to be a localized pulse with a long delay between applications of the pulse, so we enter 0 for this option. Finally, vc.exe asks for a measurement time delay, which is the additional time the Virtual Crossover waits after the waveform has been applied before it starts sampling data; this option is primarily to account for the time delay introduced by a microphone placed several feet away from a speaker under test. As a rough estimate, try entering about 1ms for every foot the microphone is away from the speaker. Also in Fig. 22 you can see the N option to invert the test waveform; if you observe that the measurement results are of opposite polarity from what you want, toggling the N option *before* uploading the test waveform will result in the opposite polarity waveform being applied to the speaker. Once a test waveform has been uploaded, return to the Main Menu and type G to start the measurement process. It will probably also be necessary to adjust the output DAC volume and muting settings from the Main Menu to achieve satisfactory measurement results. Typing Q from the Main Menu stops the measurement.

The second mechanism for setting up a measurement in Fig. 21 is to set up a sine wave. Here is one possible dialog for setting up a 1kHz sine wave after typing S in Fig. 21:



```
Enter sine wave frequency in Hz
1e3
Enter measurement delay in seconds
1_
```

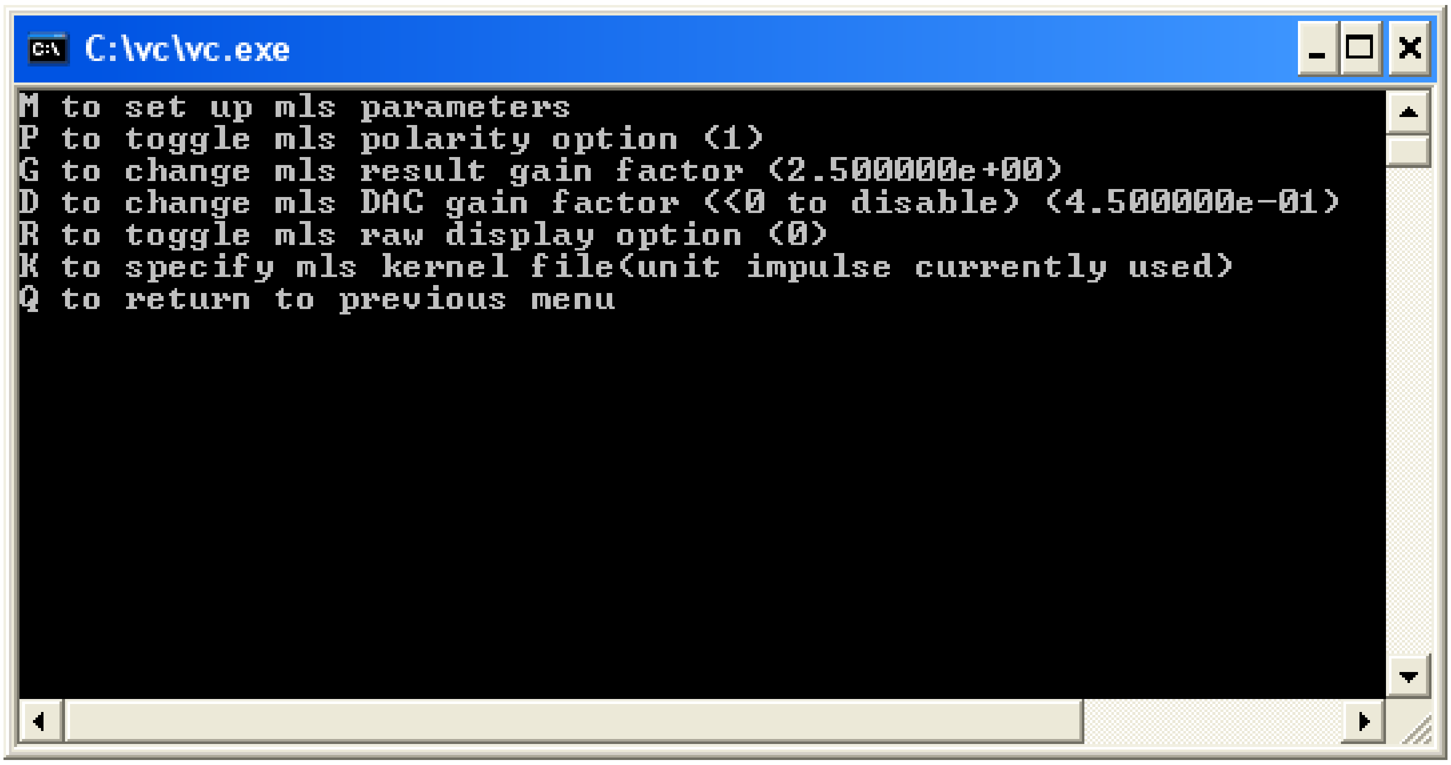
Figure 23: Setting up a 1kHz sine wave using the S option.

Sine waves in vc.exe are primarily intended for THD (Total Harmonic Distortion) measurements, which are carried out using FFTs of the measured data. For the FFT to be accurate, it is important that exactly an integral number of cycles minus one point is contained within each FFT calculation. When the user requests 1kHz, the vc.exe program figures out the closest frequency to 1kHz such that an integral number of cyles (minus one point) will be contained in the current number of measurement points. So in general, you may not end up with exactly the frequency that was requested. The frequency that vc.exe decides upon is displayed on the screen after a sine wave has been set up.

The measurement delay for a sine wave is the time delay between the start of the sine wave and when capture of measurement data is started. For a THD measurement, this can be thought of as a suitable time for the system under test to reach a steady-state condition before the measurement is begun. It is also possible to make the Virtual Crossover operate as a sine wave generator by specifying a very long delay for this parameter, for example 200 seconds.

Once a sine wave has been set up, returning to the Main Menu and typing G starts the sine wave measurement. Again adjustment of volume and muting settings will be necessary to achieve satisfactory results. After the specified time delay, the measurement automatically stops after capturing a screenful of data; or Q may also be typed to stop the sine wave measurement.

The final mechanism for setting up a measurement that is built into vc.exe is for MLS or Maximal Length Sequence measurements. Typing M from Fig. 21 brings up the following screen:



```
M to set up mls parameters
P to toggle mls polarity option (1)
G to change mls result gain factor (2.500000e+00)
D to change mls DAC gain factor (<0 to disable) (4.500000e-01)
R to toggle mls raw display option (0)
K to specify mls kernel file(unit impulse currently used)
Q to return to previous menu
```

Figure 24: Option M for setting up MLS measurements.

The details of setting up the MLS sequence are entered using the M option, which will be explained below. MLS sequences are primarily used to obtain the impulse response of the device under test (however by specifying a kernel other than the default unit impulse, the response to other waveforms can be obtained). The MLS sequence itself seems essentially like noise, but it actually has an overall periodicity. The impulse response is obtained through a cross-correlation between the measurement results and the input MLS sequence. The Virtual Crossover can either display the raw data (before the cross-correlation, primarily useful for setting volume levels) or the results for the impulse response after the cross-correlation. For MLS measurements, it is important to set the volume levels so that the Virtual Crossover input is not clipped, otherwise nonlinear distortion will appear in the measurement results which is the fault of the measurement itself, and not the device under test.

The P option in Fig. 24 controls the polarity of the MLS sequence. If you find that you end up with a negative-going impulse response, toggling the P option will result in a positive-going one.

The G option specifies the gain that is applied after the cross-correlation is carried out; it has nothing to do with the volumes of the raw signals used to carry out the MLS measurement. The latter must be changed either using the DAC volume settings from the Main Menu, or the D option described below. The gain specified by the G option is basically used to scale the impulse response result so that it will be visible on screen plots for the current vertical axis settings.

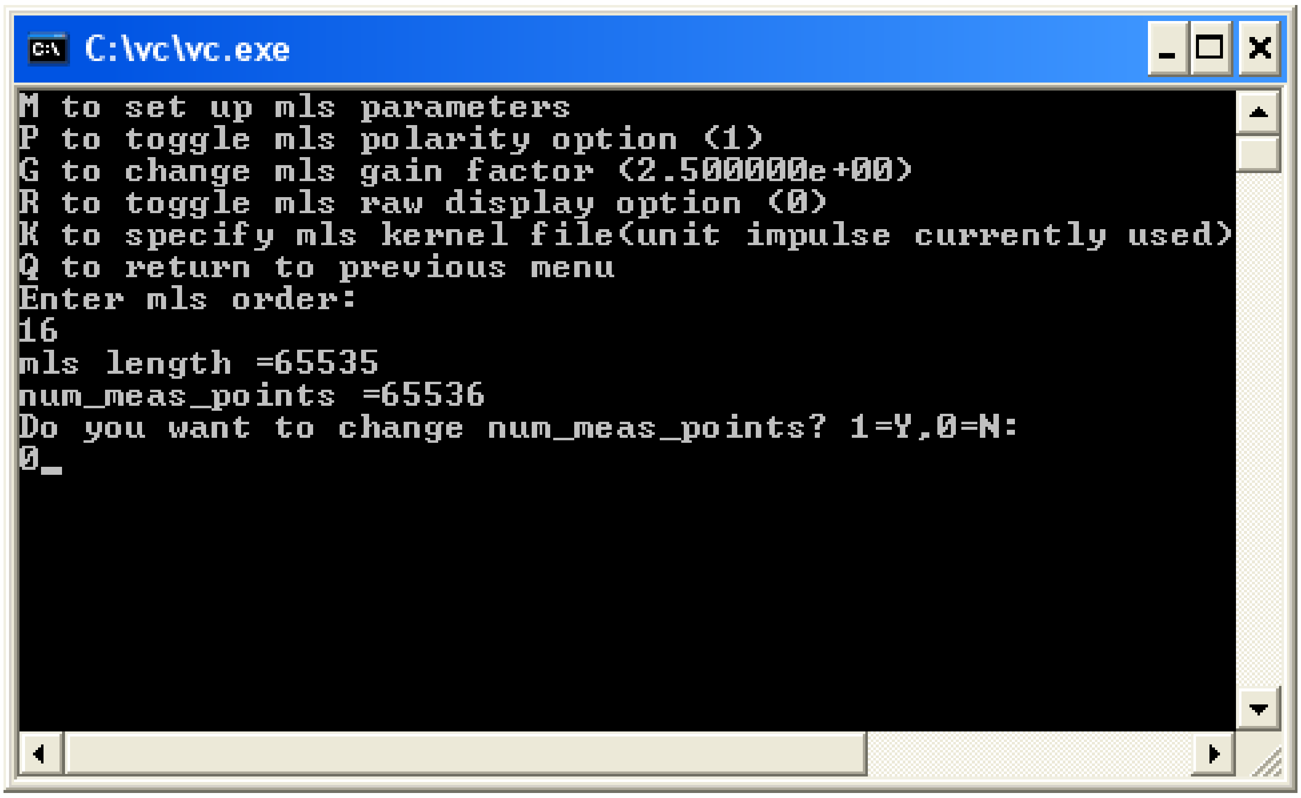
The D option sets a parameter that scales the MLS sequence sent to the Virtual Crossover for measurement output. Setting this parameter to a negative value disables the option, i.e. there is no scaling; it is also equivalent to setting the parameter to 1. This parameter is important because it is possible to overflow the range of the DAC with the MLS sequence if it is too big, which results in severe distortion of the impulse response. It is also possible to overflow the ADC range, which likewise results in severe distortion. One useful technique to avoid such distortion is the loopback test, i.e. connecting the measurement output channel directly to the input and carrying out a MLS measurement. If the DAC gain is set to the maximum value (1023) and if the raw MLS data is observed (see the R option below), I have found that if the MLS waveform does not go outside of the ±0.8 range, there is probably no overloading of the DAC.

Another way to detect possible clipping is to turn the raw option off and examine the loopback impulse response; if you see lots of random garbage in the impulse response, it is likely that you are overloading either the DAC or the ADC. To check for clipping of the ADC, monitor the raw MLS data in the actual measurement; it should stay well within the ±1.0 range.

The R option controls whether raw MLS measurement data will be displayed on time-domain plots, i.e. the data before cross-correlation is performed to obtain the impulse response. This option is primarily useful for setting DAC volume levels to values that will not clip the Virtual Crossover inputs, which can result in non-linear distortion appearing in the impulse response result.

As previously mentioned, it is possible to specify an alternate kernel for the MLS sequence besides the default unit impulse. This can be done using the K option in Fig. 24. If you type K, the program will ask for a filename where it can read in the waveform to be used as the MLS kernel. This waveform must conform to standard format for time-domain waveforms.

Typing M in Fig. 24 brings about the following dialog for setting the details of the MLS sequence:



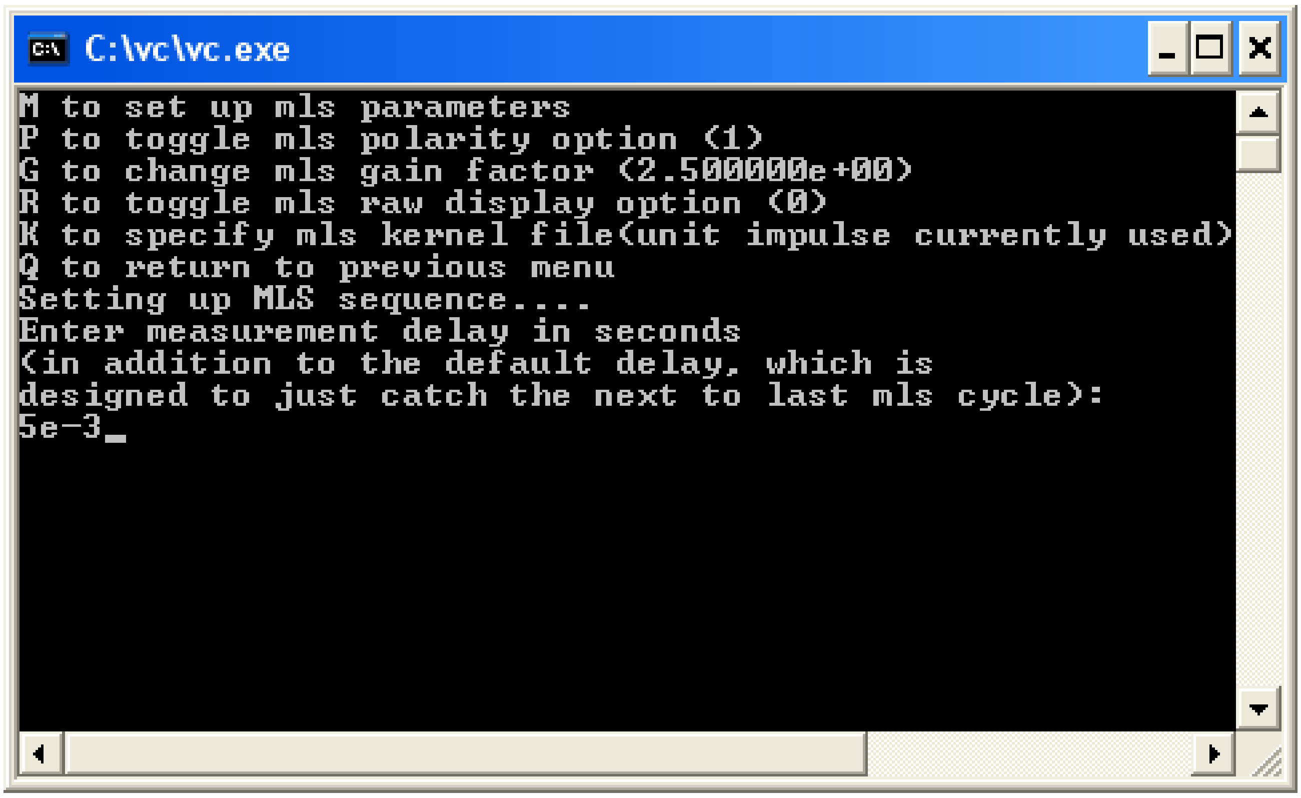
```
M to set up mls parameters
P to toggle mls polarity option (1)
G to change mls gain factor (2.500000e+00)
R to toggle mls raw display option (0)
K to specify mls kernel file(unit impulse currently used)
Q to return to previous menu
Enter mls order:
16
mls length =65535
num_meas_points =65536
Do you want to change num_meas_points? 1=Y,0=N:
0_
```

Figure 25: Option M for setting up specific MLS parameters.

First the program asks for the order of the MLS sequence that you want. In general, an MLS sequence of order n contains $2^n - 1$ points. The program vc.exe has a catalog of MLS sequences up to order 32, however the program variable storage is probably not sufficient in its current state to handle such a huge sequence; the maximum order that I have routinely used is 16.

A quantity that is related to the MLS order is the variable num_meas_points, which is the current number of measurement result values that are stored. To carry out a meaningful cross-correlation, we need to save at least one cycle of the MLS sequence. The question in Fig. 25 regarding num_meas_points gives you the opportunity to change this value. For example, it is entirely possible for the default number of measurement points in vc.exe to be 32768 when you try to set up an MLS sequence of order 16, which requires 65535 points. In this particular case we were ok, because the number of measurement points was already 65536. In general, I always set the number of measurement points to one plus the number of points required for the MLS sequence (num_meas_points must be of the form 2^n to be able to use FFT calculations).

After answering the question regarding number of measurement points, the following dialog results:



```
M to set up mls parameters
P to toggle mls polarity option (1)
G to change mls gain factor (2.500000e+00)
R to toggle mls raw display option (0)
K to specify mls kernel file(unit impulse currently used)
Q to return to previous menu
Setting up MLS sequence....
Enter measurement delay in seconds
(in addition to the default delay, which is
designed to just catch the next to last mls cycle):
5e-3_
```

Figure 26: Second stage in setting up specific MLS parameters.

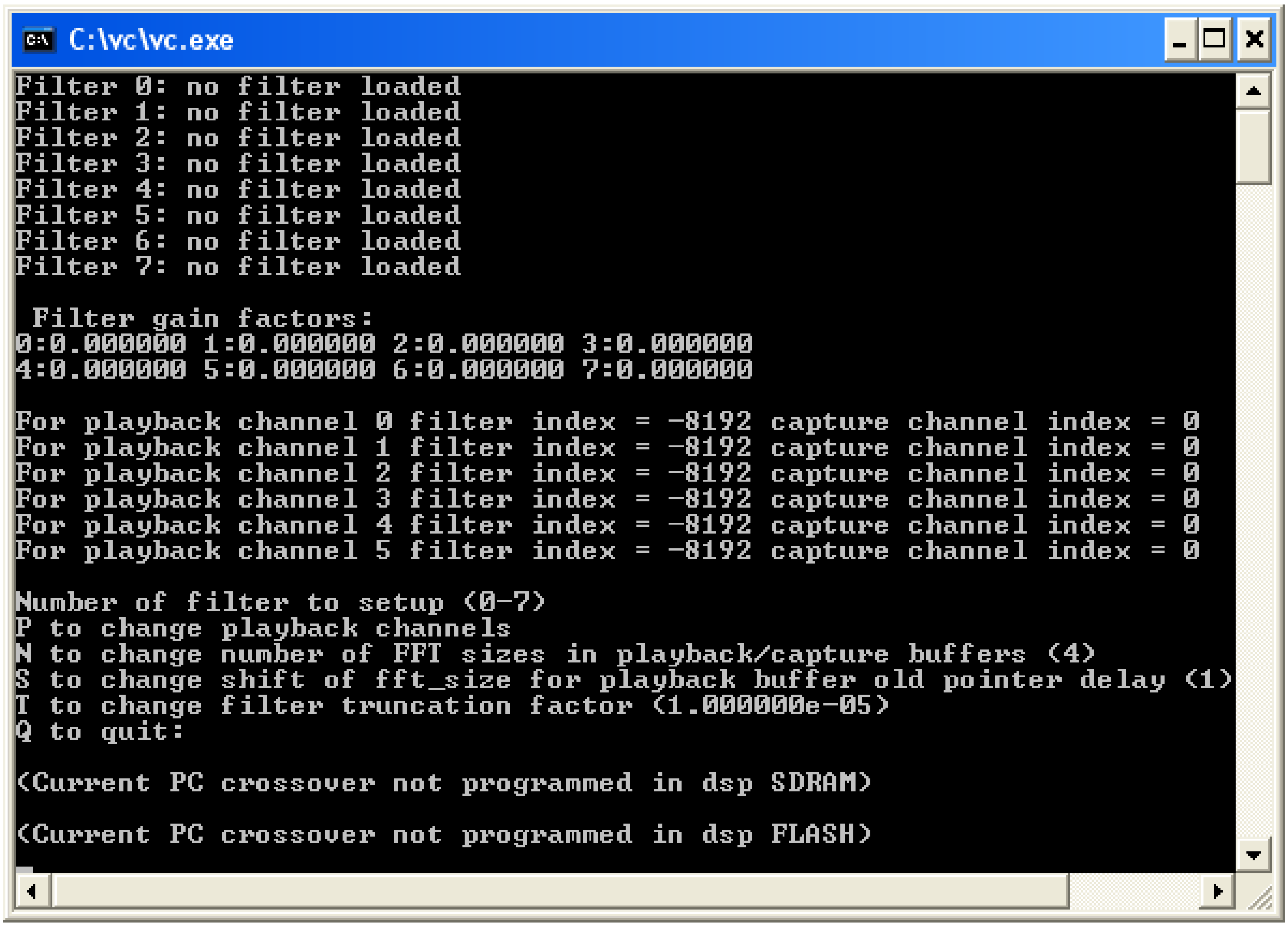
The program first sets up the actual MLS sequence, which may take a while for the larger sequences. You are then asked to enter a measurement delay. This is similar to the delay entered when we uploaded uimp_96khz using the U option, it is primarily to compensate for delay introduced by the placement of a microphone some distance away from a speaker under test. If you find that the impulse response you obtain is not localized at the beginning of the time-domain plot, adjustment of this delay can control the location of the impulse response; however you must re-load the MLS sequence each time you want to change this delay.

The overall sequence for MLS measurements is as follows. A number of MLS sequence periods is first applied to the device under test before measurement data is captured (this number of cycles is controlled by a parameter in the vc.in file). The Virtual Crossover starts taking measurement results during the next to last MLS cycle that is applied. However it waits for the additional delay specified in Fig. 26 before starting to take measurement result data.

Once an MLS sequence has been set up, typing G from the Main Menu starts the MLS measurements, which continue until Q is typed.

10 L To Setup Crossover Filters from Main Menu

Typing L from the Main Menu in Fig. 4 brings up the following screen (assuming that no crossover has been loaded yet):



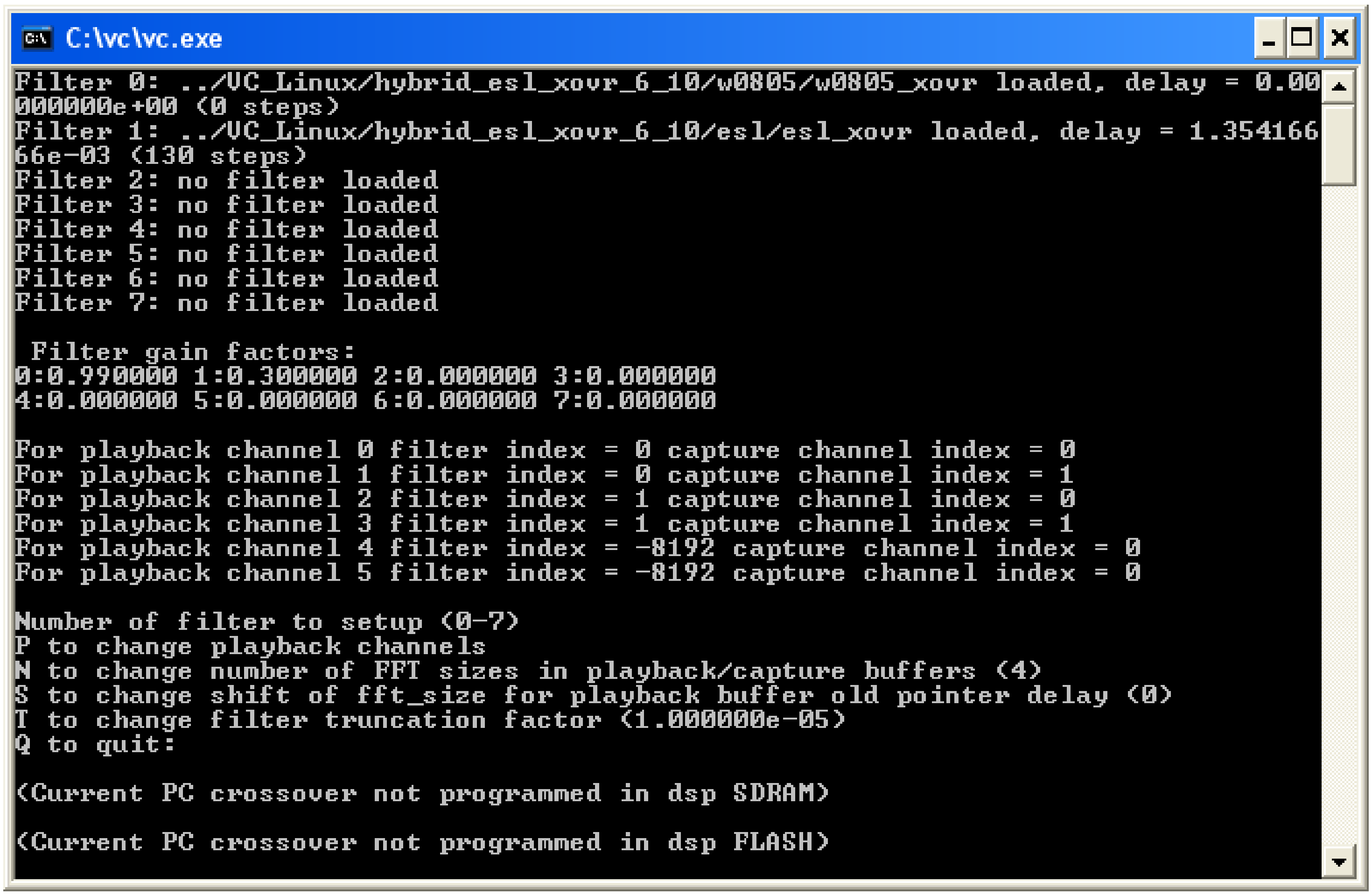
C:\vc\vc.exe

```
Filter 0: no filter loaded
Filter 1: no filter loaded
Filter 2: no filter loaded
Filter 3: no filter loaded
Filter 4: no filter loaded
Filter 5: no filter loaded
Filter 6: no filter loaded
Filter 7: no filter loaded

 Filter gain factors:
0:0.000000 1:0.000000 2:0.000000 3:0.000000
4:0.000000 5:0.000000 6:0.000000 7:0.000000

For playback channel 0 filter index = -8192 capture channel index = 0
For playback channel 1 filter index = -8192 capture channel index = 0
For playback channel 2 filter index = -8192 capture channel index = 0
For playback channel 3 filter index = -8192 capture channel index = 0
For playback channel 4 filter index = -8192 capture channel index = 0
For playback channel 5 filter index = -8192 capture channel index = 0

Number of filter to setup (0-7)
P to change playback channels
N to change number of FFT sizes in playback/capture buffers (4)
S to change shift of fft_size for playback buffer old pointer delay (1)
T to change filter truncation factor (1.000000e-05)
Q to quit:

(Current PC crossover not programmed in dsp SDRAM)

(Current PC crossover not programmed in dsp FLASH)
```

Figure 27: Option L from Main Menu to set up crossover filters.

At the top of the screen, this display shows that none of the eight possible available crossover filters has been loaded with anything yet. (It should be emphasized that the display in Fig. 27 represents the crossover information currently in vc.exe, the PC program, not necessarily what is programmed in the Virtual Crossover). Note that there are six playback channels listed in Fig. 27, whereas there are up to eight possible playback channels for the Virtual Crossover. The number of implemented playback channels is controlled by a parameter in the vc.in file. Since the maximum number of playback channels I ever require for my speaker systems is six (I only have up to three-way speaker systems), I have set this parameter to implement six channels in my vc.in files. The playback channels listed in Fig. 27 can each be assigned to a particular filter number, which will be the filter used to process input data before output to that particular output channel. Note in Fig. 27 that all the playback channels are assigned to filter index -8192. This is a convention used in the Virtual Crossover, that if a playback channel is not associated with a crossover filter, i.e. if input data is to be passed directly to the output channel without any filtering, its filter index should be specified as a negative number, minus the default FFT size.

If instead I had previously loaded my hybrid_esl_xovr_6_10 crossover from a file (Main Menu option X), typing L would yield the following display:



```
Filter 0: ../VC_Linux/hybrid_esl_xovr_6_10/w0805/w0805_xovr loaded, delay = 0.00
000000e+00 (0 steps)
Filter 1: ../VC_Linux/hybrid_esl_xovr_6_10/esl/esl_xovr loaded, delay = 1.354166
66e-03 (130 steps)
Filter 2: no filter loaded
Filter 3: no filter loaded
Filter 4: no filter loaded
Filter 5: no filter loaded
Filter 6: no filter loaded
Filter 7: no filter loaded

 Filter gain factors:
0:0.990000 1:0.300000 2:0.000000 3:0.000000
4:0.000000 5:0.000000 6:0.000000 7:0.000000

For playback channel 0 filter index = 0 capture channel index = 0
For playback channel 1 filter index = 0 capture channel index = 1
For playback channel 2 filter index = 1 capture channel index = 0
For playback channel 3 filter index = 1 capture channel index = 1
For playback channel 4 filter index = -8192 capture channel index = 0
For playback channel 5 filter index = -8192 capture channel index = 0

Number of filter to setup (0-7)
P to change playback channels
N to change number of FFT sizes in playback/capture buffers (4)
S to change shift of fft_size for playback buffer old pointer delay (0)
T to change filter truncation factor (1.000000e-05)
Q to quit:

(Current PC crossover not programmed in dsp SDRAM)

(Current PC crossover not programmed in dsp FLASH)
```

Figure 28: Option L from Main Menu with crossover already loaded.

This crossover is for a two-way speaker system, and as you can see from Fig. 28 there are now two filters defined, filters number 0 and 1. (When filters are defined, always start at number 0 and increment the filter index by 1 for each new filter defined. Jumping ahead by more than 1 will result in incorrect operation). To define a filter, enter the index for the filter; the program asks for the filename containing the filter, which must conform to standard format for time-domain files. It then asks for the time delay associated with the filter, expressed as an integer number of sampling rate intervals. It finally asks for a gain associated with the filter. (If a filter for that index has already been defined, the program first asks if you want to replace the filter with a new waveform. If you specify 0 (no) to this request, a new filter file is not read, but the program just reads in new delay and gain values for the filter already loaded).

Note in Fig. 28 that there are now two filter gains that have been defined, one for each of the two filters that have been loaded.

The playback channel information in Fig. 28 will next be explained. Since this is a two-way speaker system and there are two channels for stereo, a total of 4 playback channels are required, which are assigned index 0 through 3 in Fig. 28. Note that there is a discrepancy between the starting index for playback channels in the vc.exe program, which starts at 0, and the labeling on the back of the Virtual Crossover, which starts at playback channel 1. The mapping between playback channels in Fig. 28 and playback channels on the Virtual Crossover back panel is as follows:

PC playback channel 0 → VC playback channel 1L
PC playback channel 1 → VC playback channel 1R
PC playback channel 2 → VC playback channel 2L
PC playback channel 3 → VC playback channel 2R

PC playback channel 4 → VC playback channel 3L
PC playback channel 5 → VC playback channel 3R
PC playback channel 6 → VC playback channel 4L
PC playback channel 7 → VC playback channel 4R

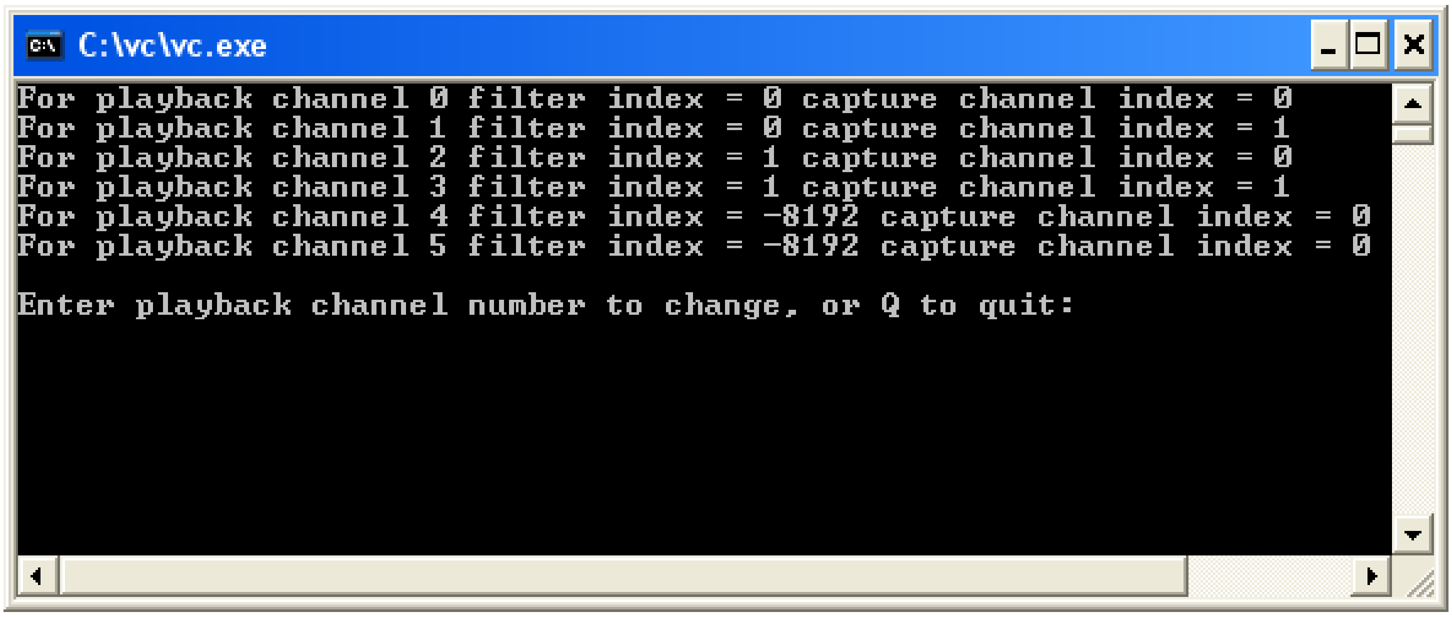
where for example "1L" on the Virtual Crossover back panel means output channel 1 Left. With this mapping in mind, the playback channel assignments in Fig. 28 mean that the left and right woofer drivers (associated with the w0805_xovr filter) would be connected to the 1L and 1R Virtual Crossover outputs, respectively; and that the left and right ESLs (associated with the esl_xovr filter) would be connected to the 2L and 2R Virtual Crossover outputs.

The mapping between PC program capture channels (starting at index 0) and Virtual Crossover input channels is:

PC capture channel 0 → VC input channel Input-L
PC capture channel 1 → VC input channel Input-R

So you can see from Fig. 28 that the 1L and 2L Virtual Crossover outputs have the left input channel as their source, and the 1R and 2R outputs have the right input channel as their source. (For SPDIF input, the Input-L and Input-R inputs become the left and right channels of the SPDIF input stream).

Playback channel assignments can be set up by typing P from the Setup Crossover Filters Menu in Fig. 28; the following sub-menu is displayed, assuming that the hybrid_esl_xovr_6_10 crossover has already been loaded:



```
For playback channel 0 filter index = 0 capture channel index = 0
For playback channel 1 filter index = 0 capture channel index = 1
For playback channel 2 filter index = 1 capture channel index = 0
For playback channel 3 filter index = 1 capture channel index = 1
For playback channel 4 filter index = -8192 capture channel index = 0
For playback channel 5 filter index = -8192 capture channel index = 0

Enter playback channel number to change, or Q to quit:
```

Figure 29: Option P for Change Playback Channels Menu.

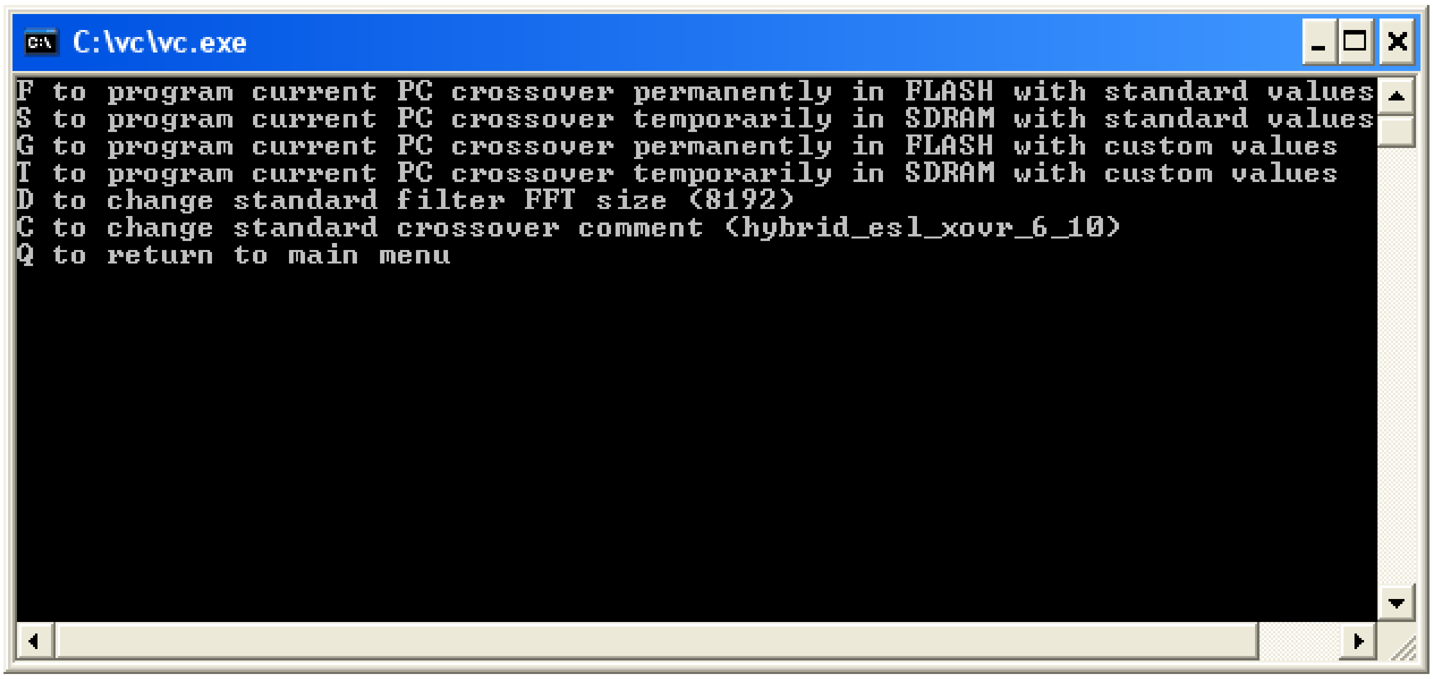
To change a playback channel's assignments, enter the index associated with the playback channel. The program then asks the filter index to be associated with the playback channel, and the capture channel index to be the source for that playback channel.

The N and S options in Fig. 28 have to do with the buffering scheme used by the Virtual Crossover to process input data through the crossover filters. The basic concern here is that we don't want to overwrite any part of the dsp's buffers that have not been used yet, and similarly we want to have output data available when it is needed. Option N controls the size of the playback and capture buffers used in the Virtual Crossover, and option S controls a playback buffer pointer setting that ensures there will be sufficient time to calculate new playback data by the time it is needed. These values can also be initialized in the vc.in file. You may never need to change these values, but the mechanism for changing them is provided in case you should encounter playback errors using the Virtual Crossover.

The T option, or truncation factor, is used to limit the size of the crossover filters that are read in. Let $M_{\max}$ denote the maximum magnitude for a particular crossover filter that is loaded. If all filter values beyond a certain time, denoted by $t_{\max}$, are less in magnitude than $T \times M_{\max}$, then only filter values up to $t_{\max}$ are included in the filters loaded into the Virtual Crossover. This option can effectively be defeated by setting the truncation factor to an extremely small number.

11 R Option from Main Menu to Program DSP With Current Crossover

Typing R from the main menu brings up the following sub-menu:



```
F to program current PC crossover permanently in FLASH with standard values
S to program current PC crossover temporarily in SDRAM with standard values
G to program current PC crossover permanently in FLASH with custom values
T to program current PC crossover temporarily in SDRAM with custom values
D to change standard filter FFT size (8192)
C to change standard crossover comment (hybrid_esl_xovr_6_10)
Q to return to main menu
```

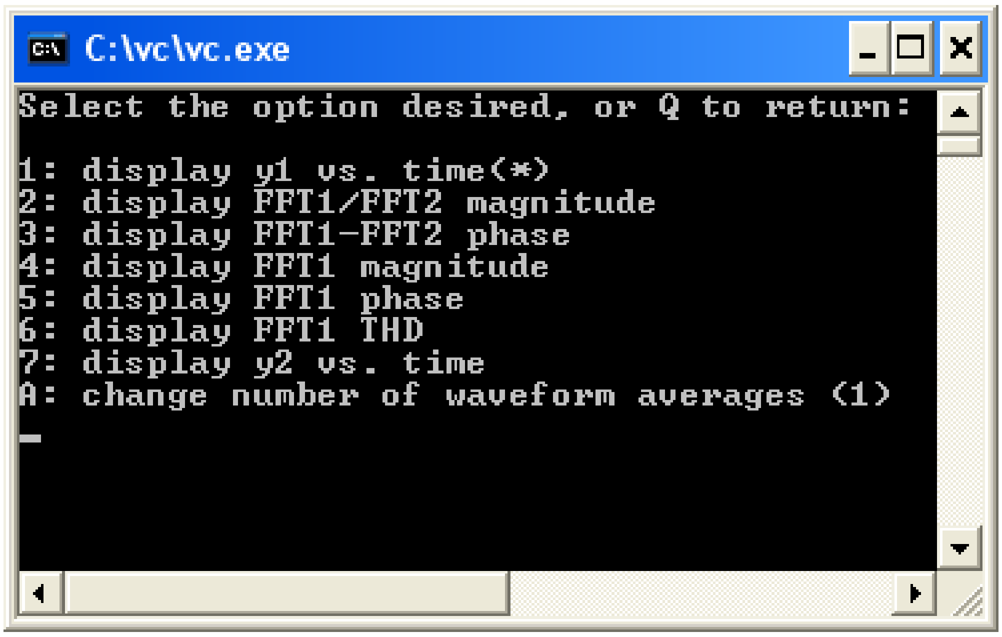
Figure 30: Option R for Program DSP With Current Crossover Menu.

This menu allows you to program either the FLASH or SDRAM memory of the Virtual Crossover with the current crossover that is set up on the PC. This crossover may have been set up in several steps using the L option from the Main Menu, or it may have been loaded in one step from a previously set up crossover using the X option from the Main Menu. If you are in the process of developing a crossover and are not sure yet if the crossover is optimum, loading it into SDRAM memory will allow you to try it out (either through measurements or by playing music) without permanently programming the crossover into FLASH memory. On the other hand, if the crossover is in good shape and you want the Virtual Crossover to come up with it in memory the next time it is powered up, you can choose to program the crossover into FLASH memory. (Keep in mind, though, that the Virtual Crossover uses the crossover that is currently in SDRAM, so if you program only FLASH memory only it will not use the crossover until the next power-up).

I would recommend selecting either the F or S options in Fig. 30 to program either the FLASH or SDRAM memories. Before programming the Virtual Crossover, you may want to change the crossover comment using the C option; this comment will appear in the vc.exe program as a reminder of which crossover is currently loaded in the dsp. You can also change the standatd FFT size using the D option; 8192 is the maximum size that can be accommodated by the Virtual Crossover. After the crossover comment and FFT size are what you want, pressing either F or S will start the programming process.

12 Main Menu Option A to Change Automatic Mode

Typing A from the Main Menu in Fig. 4 brings up the Change Automatic Mode Menu:



```
Select the option desired, or Q to return:

1: display y1 vs. time(*)
2: display FFT1/FFT2 magnitude
3: display FFT1-FFT2 phase
4: display FFT1 magnitude
5: display FFT1 phase
6: display FFT1 THD
7: display y2 vs. time
A: change number of waveform averages (1)
_
```

Figure 31: Change Automatic Mode Menu.

If you have set up a measurement by either uploading a waveform (U option in Fig. 21) or by setting up an MLS sequence (option M), pressing G from the Main Menu will start the measurement process. The measurements will continue periodically until Q is pressed, and the results of the measurements will be displayed on the PC screen. This periodic updating of measurements on the PC is referred to as Automatic Mode, since it proceeds without any further intervention by the user. Exactly what is displayed, however, can be set up using the Change Automatic Mode Menu of Fig. 31.

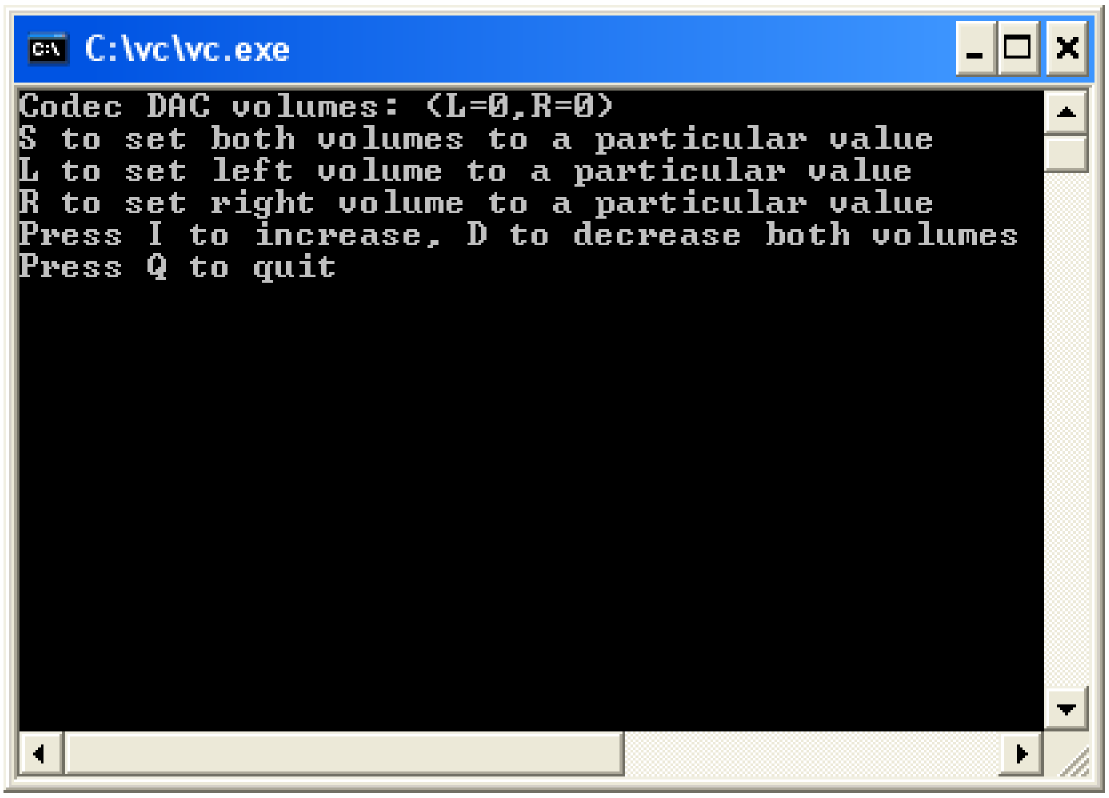
By default, the vc.exe program starts out by displaying time-domain waveform 1 on the screen. (The currently selected automatic mode is indicated by the asterisk in Fig. 31). However, you may be more interested in looking at the magnitude of the frequency response for the measurement; selecting option 4 will display the FFT magnitude for waveform 1. If you have also loaded a waveform 2 to be used as a standard or reference for the measurement, it might be preferable to look at the ratio of the FFT1 and FFT2 magnitudes, which can be selected using option 2. Phase response can also be displayed by selecting options 3 or 5. There is also an option to automatically update THD (Total Harmonic Distortion) results (although I must confess that I've never selected this option so I don't know if it works – I am sure that individual THD measurements can be carried out, though). Entering the number of your selection moves the asterisk to the line that you have selected.

Whatever automatic mode is selected in Fig. 31, it is suggested to first set up the plot by going through the individual steps of the process one time. For example, if you want to look at the ratio FFT1/FFT2, start by first going to the FFT Menu and computing the ratio using the D option followed by plotting the magnitude of FFT1. This ensures that everything is set up as you want it to be.

The vc.exe program also has the capability of averaging the results for several measurements in a row. This is especially needed if you are doing pulse testing of loudspeakers, since any background noise can distort the results for the pulse shape. Using the A option, you can specify the number of measurement cycles the Virtual Crossover should go through before updating the measurement results. By default this value is set to 1, or no averaging. In addition to pulse testing, averaging can also be used for MLS sequences, although testing with MLS sequences is much less susceptible to noise than pulse testing is.

13 Main Menu V Option To Change DAC Volume Levels

Pressing V from the Main Menu brings up the following display:



```
Codec DAC volumes: (L=0,R=0)
S to set both volumes to a particular value
L to set left volume to a particular value
R to set right volume to a particular value
Press I to increase, D to decrease both volumes
Press Q to quit
```

Figure 32: Change DAC Volume Levels Menu.

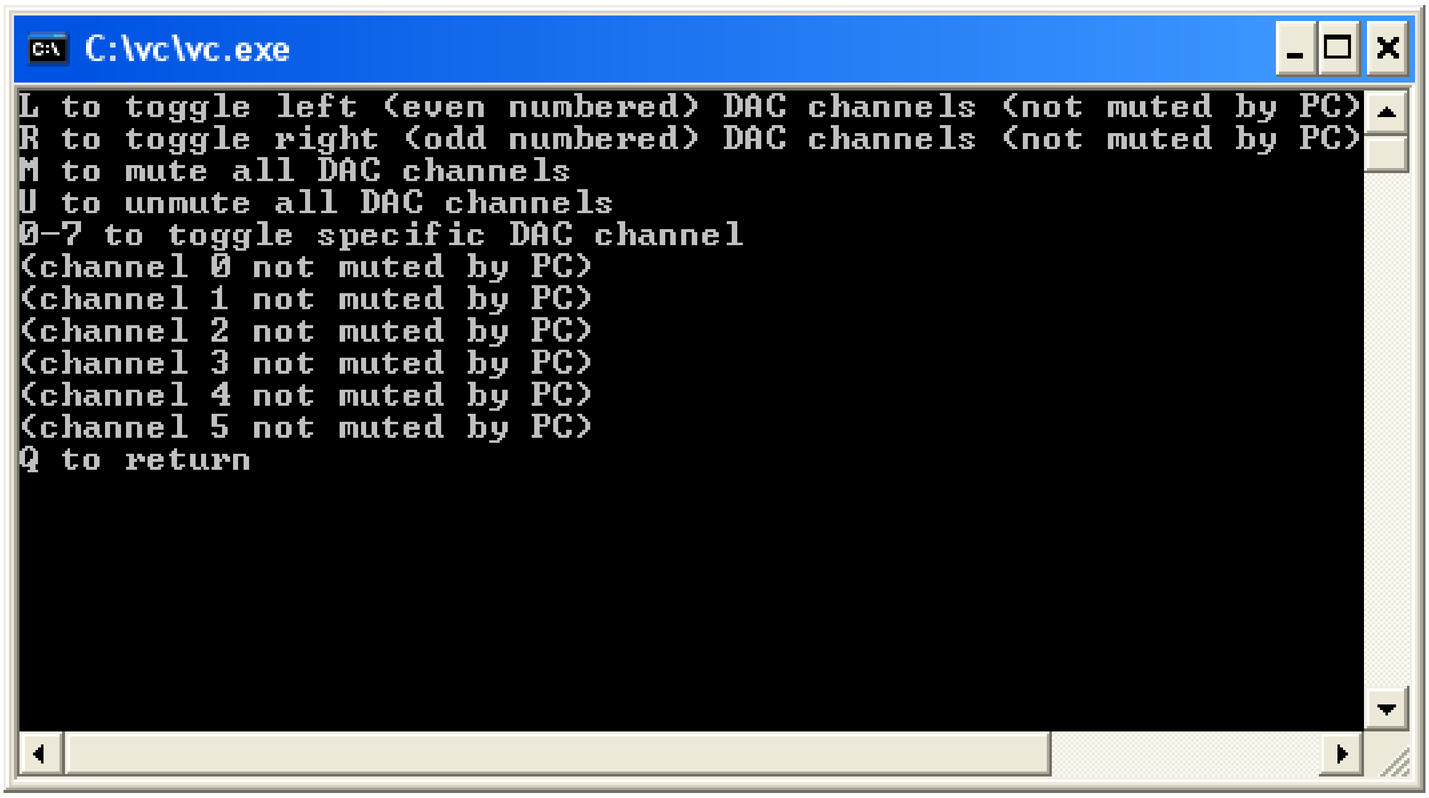
This menu can be used to set the DAC volume levels, which can assume values between 0 (muted) and 1023 (maximum output). Note that values can only be specified for the Left and Right channels. The value specified for the Left channels will apply to all output channels in the L row on the Virtual Crossover back panel (1L through 4L), and the value specified for the Right channels will apply to the R row on the Virtual Crossover back panel (1R through 4R). The exception to this rule is if a particular channel is muted, in which case it will not be affected by changes in volume.

Note also that both left and right channel volume settings can be changed in tandem from the Main Menu by pressing the uparrow and downarrow keys. Also, pressing the leftarrow and rightarrow keys from the Main Menu changes the balance between left and right channels.

Further note: the vc.exe program does not poll the Virtual Crossover to determine its current playback volume state, it only keeps track of volume changes that you make from the PC. Since the Virtual Crossover playback volumes can be changed independently from the PC, either by pressing the buttons on the front panel or by using the remote control, it is quite possible that the volume indication on the PC screen does not always reflect the true volume settings of the Virtual Crossover.

14 Main Menu U Option to Mute/Unmute Channels

Pressing U from the Main Menu brings up the following display that can be used to mute or unmute playback channels:



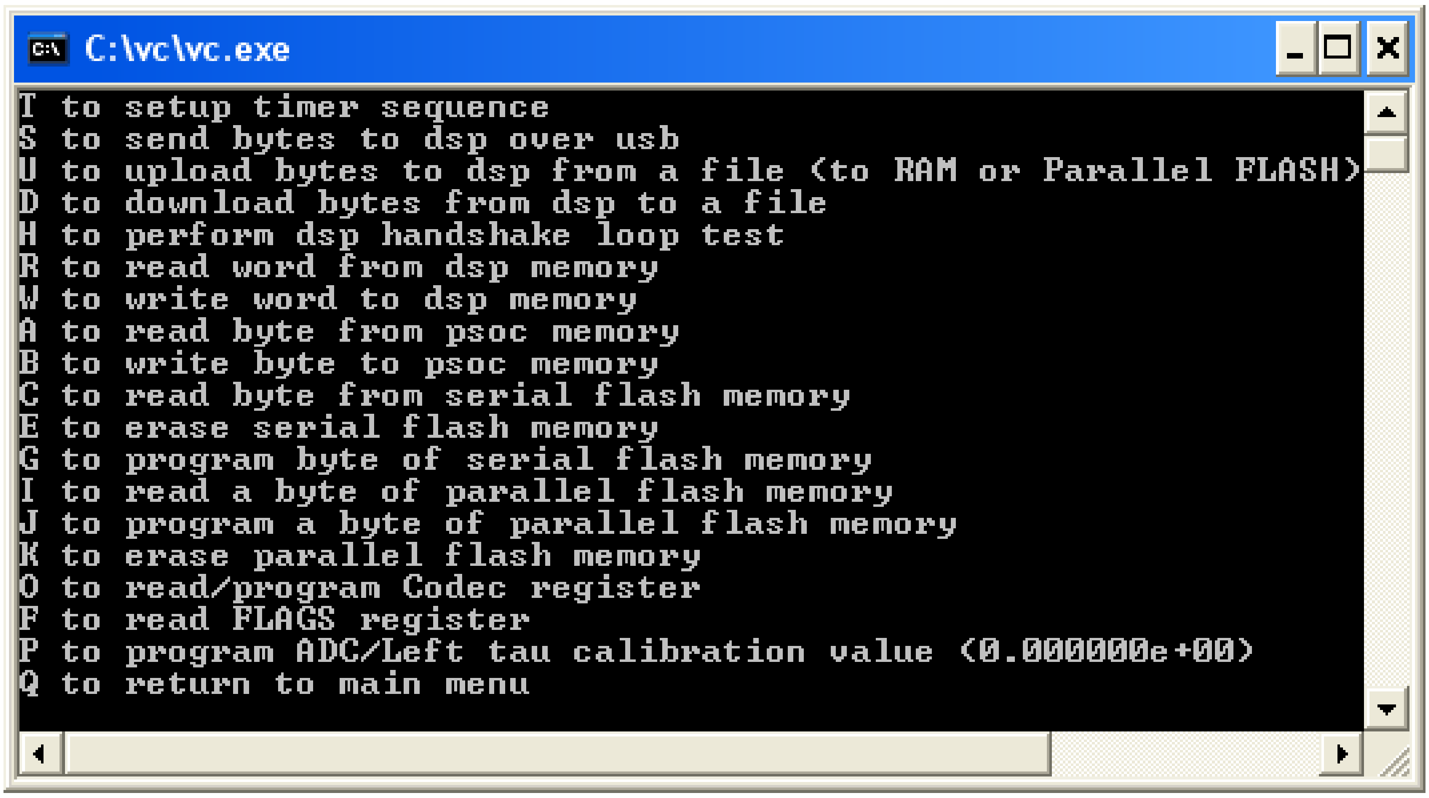
```
L to toggle left (even numbered) DAC channels (not muted by PC)
R to toggle right (odd numbered) DAC channels (not muted by PC)
M to mute all DAC channels
U to unmute all DAC channels
0-7 to toggle specific DAC channel
(channel 0 not muted by PC)
(channel 1 not muted by PC)
(channel 2 not muted by PC)
(channel 3 not muted by PC)
(channel 4 not muted by PC)
(channel 5 not muted by PC)
Q to return
```

Figure 33: Main Menu Option U To Mute/Unmute Playback Channels.

The first four options in Fig. 33 allow you to toggle the mute/unmute state for all the left or right channels separately, or for all channels. Entering the index for a particular playback channel will toggle the state for that particular channel alone. (Note that in general playback channels 0-7 can be muted or unmuted; only six channels are displayed in Fig. 33 because I implemented only six channels through the settings in my vc.in file. I only need six channels for two stereo 3-way speakers).

15 Main Menu Test/Debug Menu

Pressing D from the Main Menu brings up the Test/Debug Menu:



```
T to setup timer sequence
S to send bytes to dsp over usb
U to upload bytes to dsp from a file (to RAM or Parallel FLASH)
D to download bytes from dsp to a file
H to perform dsp handshake loop test
R to read word from dsp memory
W to write word to dsp memory
A to read byte from psoc memory
B to write byte to psoc memory
C to read byte from serial flash memory
E to erase serial flash memory
G to program byte of serial flash memory
I to read a byte of parallel flash memory
J to program a byte of parallel flash memory
K to erase parallel flash memory
O to read/program Codec register
F to read FLAGS register
P to program ADC/Left tau calibration value (0.000000e+00)
Q to return to main menu
```

Figure 34: Main Menu Option D For Test/Debug Menu.

The features in this menu were mostly used during the debugging stage in the Virtual Crossover development, and they are not very useful unless you know the details about dsp variable addresses and algorithms. Therefore I won't try to go into details here about how to use all the options.

There is, however, one feature that might be useful, namely the H option or handshake loop test. If you type H the program asks whether you want to send single handshakes or packets of 126-byte handshakes, and it then asks how many handshake loops to carry out in the test. This is intended to test the timing of the usb communications interface between the PC and the Virtual Crossover. So, for example, you could request thousands of handshake loops and time how long it takes to complete, then divide by the number of handshakes you requested to get the time per handshake loop, and divide by the number of bytes to determine the time required per byte of transfer.

16 Main Menu Option C To Set Codec Frequency

Typing C from the Main Menu allows you to set the Codec frequency, which is the frequency at which the Virtual Crossover inputs and outputs data (this is only the case if the Virtual Crossover is not receiving SPDIF input. If you are playing music with an SPDIF signal as input, the Virtual Crossover will lock to the frequency of the SPDIF signal). Actually, there are two different things you need to do in order to set the sampling frequency. There is a jumper on the back panel which allows you to select between two possible sets of frequencies, 48/96kHz or 44.1/88.2kHz. For example, placing the jumper in the 48/96kHz position restricts the possible Virtual Crossover frequencies to either 48kHz or 96kHz; you can select which of those frequencies you want with the C option from the Main Menu. Similarly, with the jumper in the 44.1/88.2kHz position you can only select either 44.1kHz or 88.2kHz.

Whatever selection you make with the C option will not take effect until the next time that the Virtual

Crossover is powered up.

17 Main Menu E Option To Get Underrun Report

An underrun is an error condition where the Virtual Crossover processor is required to output data but insufficient new data has been calculated for it to keep up with the demand. If this error occurs, the dsp keeps track of it in variable locations. Pressing E from the Main Menu reads the variables related to underrun errors and reports whether or not any have occurred. If they have, it is an indication that the present crossover is putting too much of a demand on the dsp processor in the Virtual Crossover.

18 Main Menu O Option To Change Automatic Play Music Mode

Unless you use this option to change its behavior, the Virtual Crossover automatically comes up in Play Music Mode when it is powered up, provided it detects that a crossover has been programmed into its memory. You can change this behavior at startup using the O option. When you type O from the Main Menu, a sub-menu appears that displays the current configuration for the Automatic Play Music Mode (i.e. whether or not it is enabled) and asks you to enter an integer (0 or 1) to identify whether or not you want this option to be enabled. You can also type Q to return to the Main Menu without changing anything, if you are satisfied with the current mode. If you do change the mode, the change will first take effect the next time that the Virtual Crossover is powered up.

You will probably want to defeat the Automatic Play Music Mode if you use the Virtual Crossover mostly for measurements. If the Virtual Crossover is powered up with the output of a microphone at its input, the potential exists for positive feedback that could reach destructive levels if the Virtual Crosssover output is connected to a power amplifier driving the speaker under test. You can also defeat the Automatic Play Music Mode by pressing the Mode button on the front panel before and during power-up, but it would be very inconvenient if you needed to do that every time you turned on your system. In that case, using the O option to defeat Automatic Play Music Mode would be preferable.

19 Measurement Examples – MLS Measurement

This section of the manual presents some examples of different types of measurements you can carry out with the Virtual Crossover. With each measurement I will give an overview of the steps for the measurement and I will present some of the results. If any of the steps I describe are unclear, please refer to the section of the manual that describes that function in greater detail.

MLS measurements can be used in many applications, for testing the acoustic response of speakers, the input impedance of speaker drivers, or the electrical response of circuits. For this example I did something very simple – I connected the 1L output of the Virtual Crossover to the Left Input. This loopback test will determine the overall impulse response of the Virtual Crossover.

The first step is to make sure that the Virtual Crossover does not have any crossover filters that it is using in SDRAM memory; we don't want to filter the MLS signal for this test, rather we want the Virtual Crossover to output the MLS sequence directly. One easy way to ensure this is to program the Virtual Crossover with the "blank" crossover that comes up in the vc.exe program when you start it up. The starting crossover configuration should be similar to the display in Fig. 27 (depending on the options that you selected in the vc.in file). Note that the filter indices are negative for all the playback channels in Fig. 27, which is the convention used in the Virtual Crossover when a playback channel is not supposed to use any filter. You can program this crossover into SDRAM memory by choosing the R option from the Main Menu. It is a good idea to change the default comment to something like "blank_crossover" before programming the dsp with the crossover, so you will be reminded later what crossover is currently in memory.

The next step is to select the M option from the Main Menu to set up a measurement, then select M again to set up an MLS sequence. For this measurement I selected an MLS sequence of order 16, and I set num_meas_points to 65536. The sampling rate for the measurement was 96kHz. I chose no additional delay for the MLS measurement (since there is no delay introduced from, for example, microphone placement). I selected the "Raw" option to first view the amplitude of the MLS signal in waveform 1 to make sure I was not clipping the input. I set the DAC volumes (V option from the Main Menu) to 512.

After the MLS sequence is set up and the Virtual Crossover 1L output is connected to Left Input, typing G from the Main Menu starts the measurement. You should see periodically updated displays on the screen as follows:

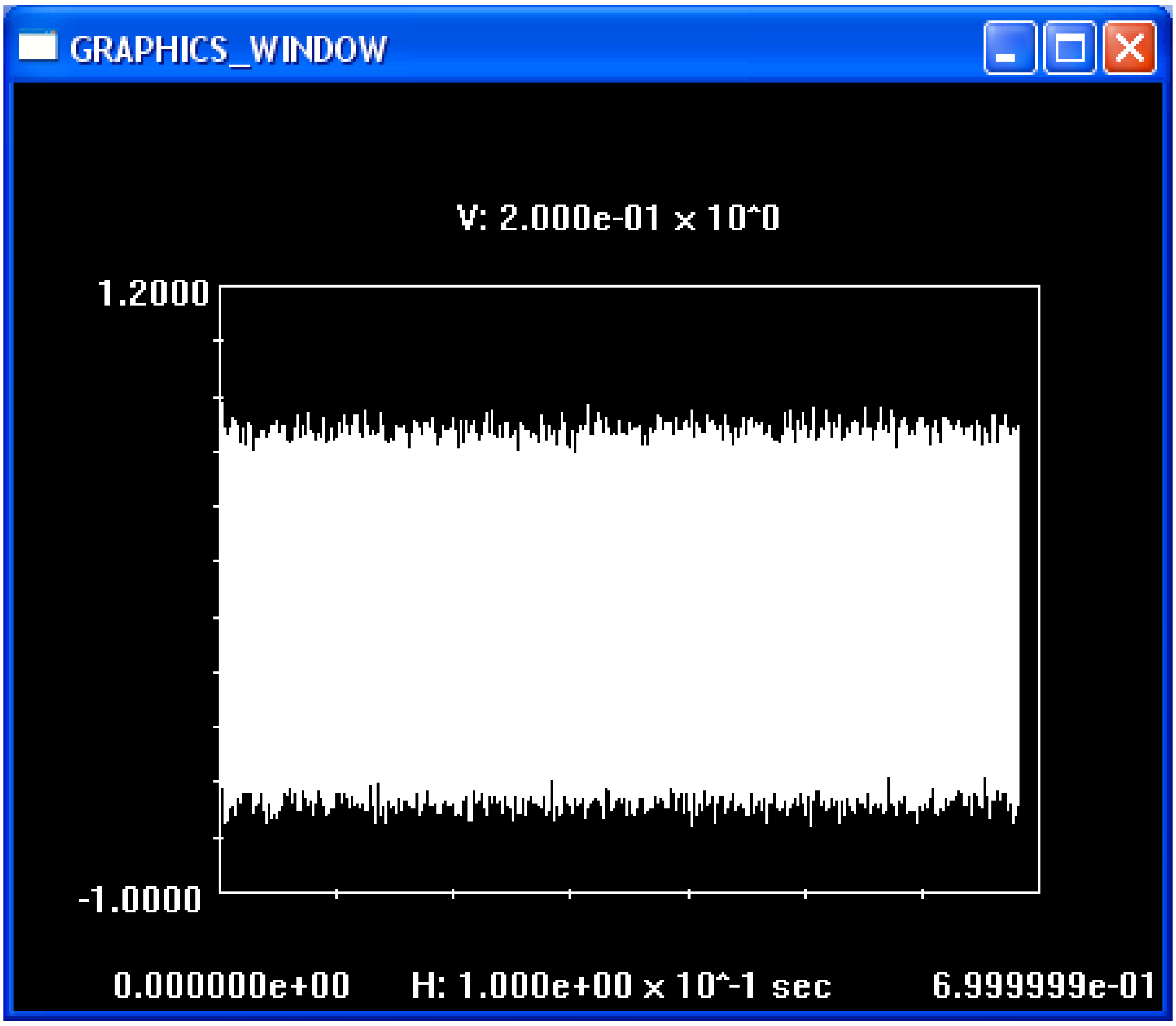


Figure 35: Waveform 1 display during MLS test with Raw option enabled.

The display in Fig. 35 appears to be ok, since none of the peaks are close to ±1. Typing Q stops the measurements. Next I went back to the Set Up A Measurement option from the Main Menu, and selected MLS Measurements again. I toggled the Raw option off, so that the vc.exe program would do the cross-correlation at each measurement to determine the impulse response. Typing G again from the Main Menu re-starts the measurements, but this time the display on the screen shows the impulse response. Note that with the default plotting parameters the impulse response will not be visible, since the plot extends for nearly 0.7 seconds and the impulse response occurs in the first couple of milli-seconds. To see the impulse response, it is first necessary to stop the measurements by typing Q, then go to the Time Waveforms Menu, and select a plot of waveform 1. Use Cursors Mode (by typing C, T for toggle and the left- and right-arrow keys, using A and B to speed up and slow down the cursor movement) to zoom in on the initial part of the curve in order to see the impulse response. You can also use the A option from the Time Waveforms Menu to better display the impulse response. After carrying out these procedures, you should end up with a display something like this:

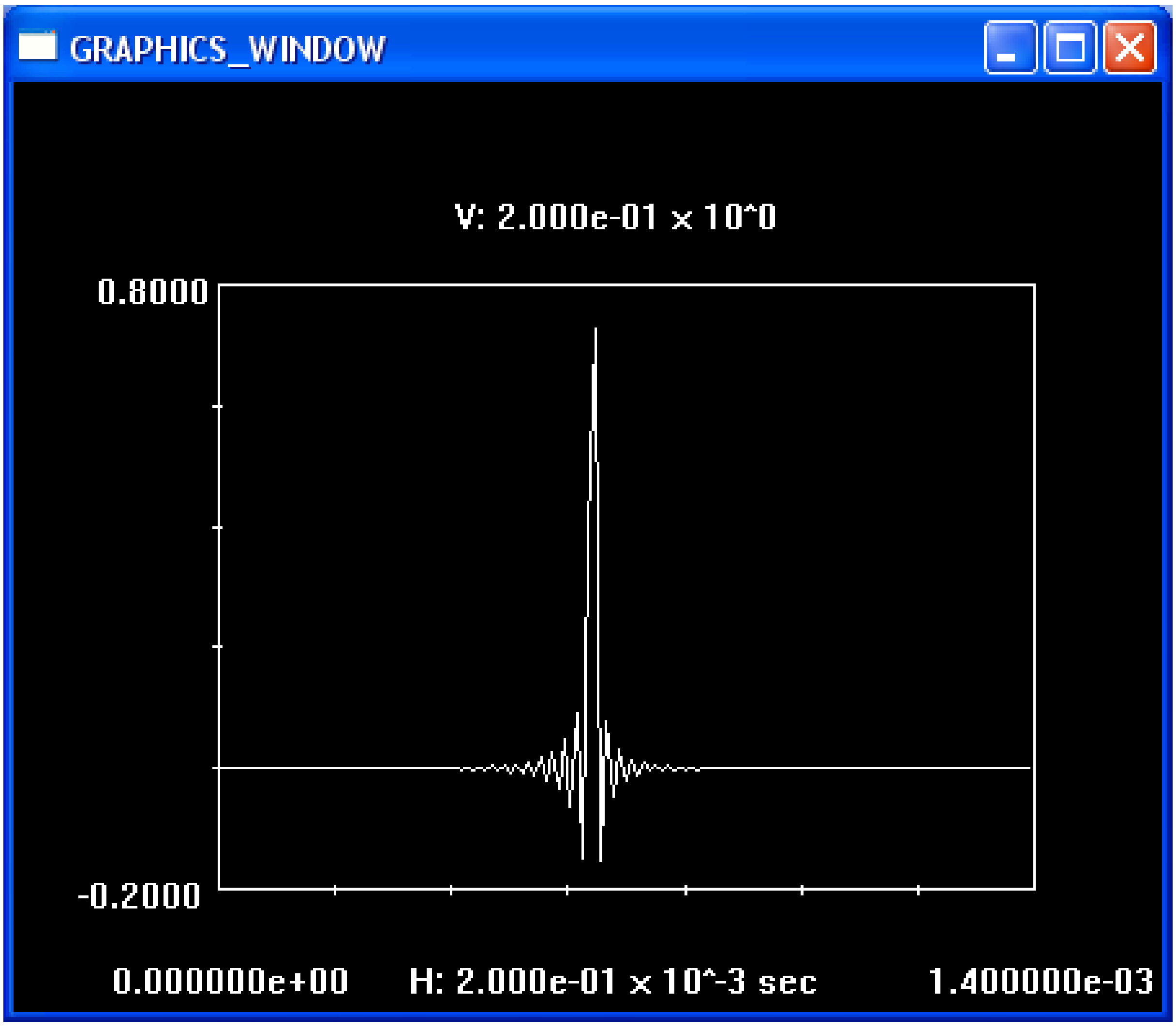


Figure 36: Impulse response of Virtual Crossover loopback from MLS measurement.

What Fig. 36 shows is really the impulse response of the Codec used in the Virtual Crossover, which serves to band-limit signals by the time we reach the folding frequency of 48kHz. We can view the frequency response by going to the FFT Menu, pressing 1 to take the FFT of waveform 1, and then typing A followed by 1 to display the frequency response of waveform 1. Here is the result I obtained:

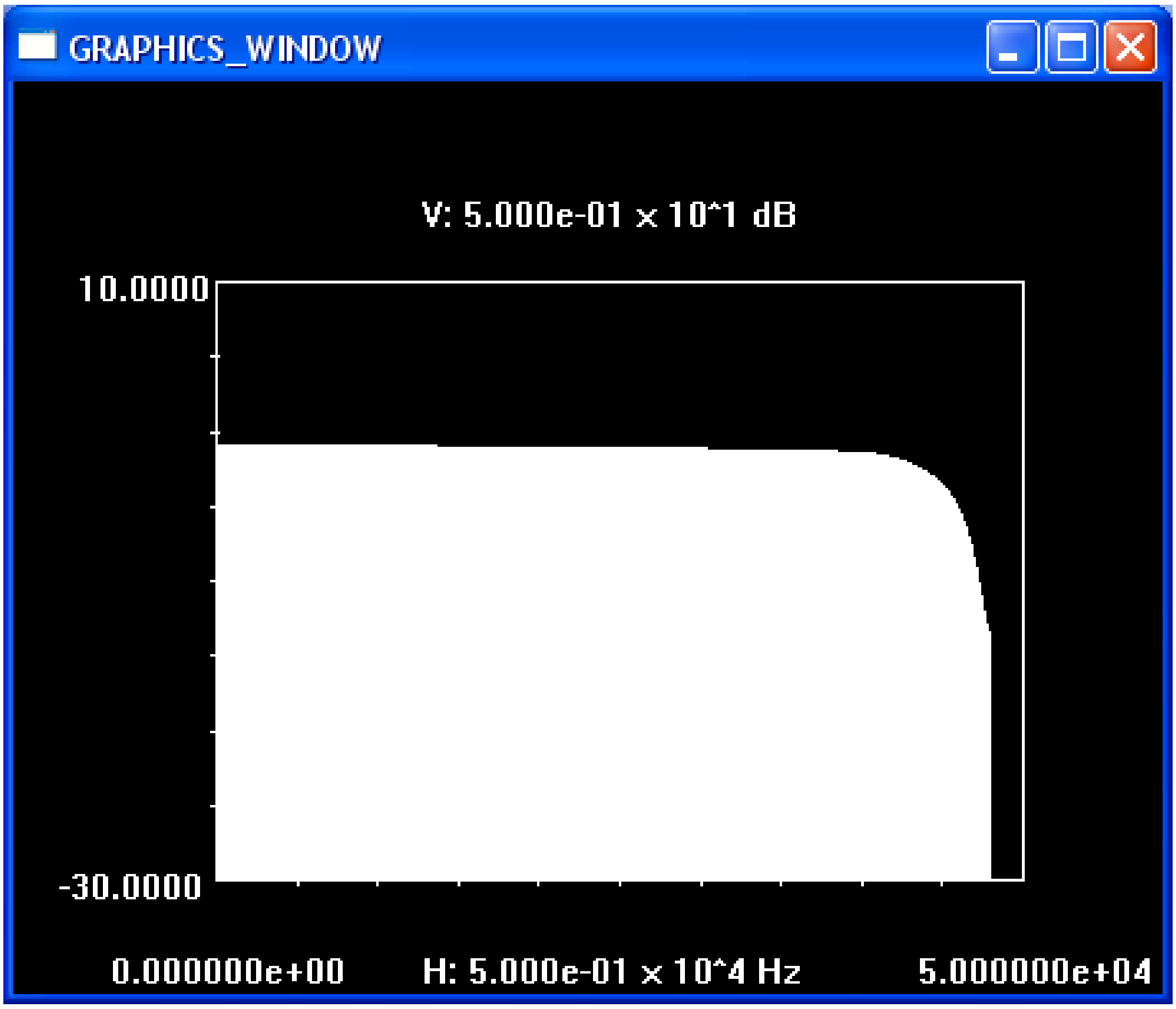


Figure 37: Frequency response of Virtual Crossover loopback from MLS measurement.

20 Measurement Examples – Sine Wave THD Measurement

I used the same loopback connection as in the previous section to measure the overall THD+N (Total Harmonic Distortion Plus Noise) for the Virtual Crossover. This quantity is defined as follows:

$$\mathrm{THD} = \frac{\sqrt{\sum_{i \neq i_{\max}} V_i^2}}{|V_{i_{\max}}|}, \tag{5}$$

where in Eq. 5 V_i denotes the i^{th} complex frequency component in the frequency-domain response, and $i_{\max}$ denotes the frequency component corresponding to the sine wave frequency. The sum in the numerator of Eq. 5 is carried out over all frequency components out to the folding frequency.

First make sure that the Virtual Crossover is not using any crossover filters in SDRAM memory (see the previous section). Type M from the Main Menu to set up a measurement, then type S to set up a sine wave measurement. I set the number of points to 65536, the sampling rate to 96kHz, and the requested sine wave frequency was 1kHz. I specified the delay for the measurement as 1 second; this is the amount of time the Virtual Crossover will wait before taking measurement results, to give the system sufficient time to reach steady state.

After setting up the sine wave, return to the Main Menu and set the DAC output volume to 1023 (V option). Then type G to start the measurements. Note that with the default plot parameters, you will not be able to see the details of the sine wave cycles since the plot will last for about 0.7 second. To see the details you will need to type Q to stop the measurements, then go to the Time Domain Waveforms Menu, and use the P option to plot waveform 1. Use Cursors Mode (C to start, T to toggle cursors, left- and right-arrow keys to move cursors, A and B to speed up and slow down cursor movement) to select a time window about 10ms wide. You should then see a plot similar to the following:

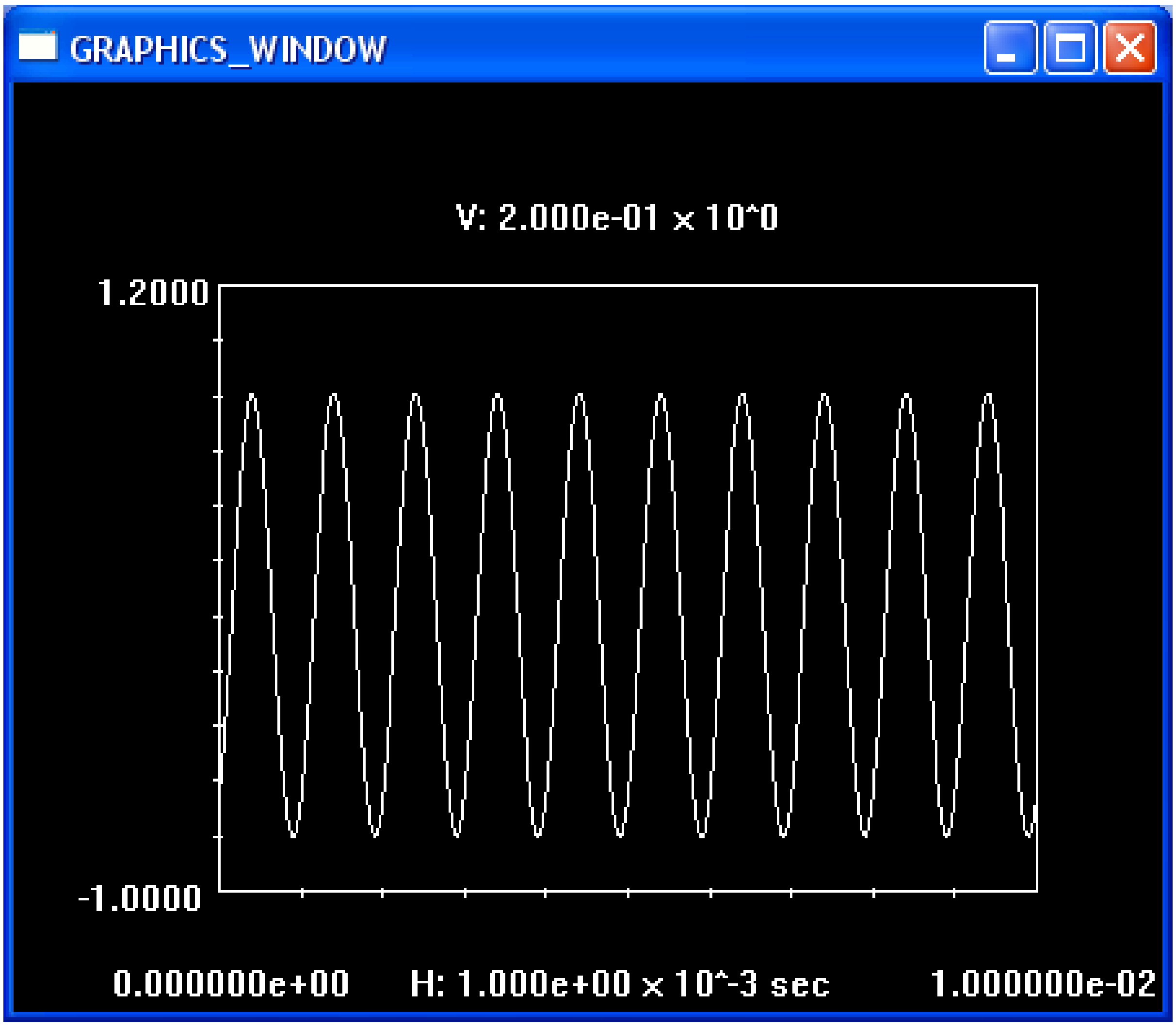


Figure 38: 1kHz loopback sine wave test of Virtual Crossover.

You may not think that the fidelity of the sine waves in Fig. 38 is very good, but remember that from sampling theory we only need to sample a waveform at a frequency greater than twice the maximum frequency in the waveform to be guaranteed to retain all the information contained in the waveform, provided it is truly band-limited.

To carry out the THD+N measurement, go to the FFT Menu and toggle the T option. Then type 1 so that FFT1 is calculated, and choose the A option to plot the results, specifying channel 1. Since this plot requires a wide dynamic range, you will need to adjust the vertical scale using the P option also. I chose a maximum amplitude of 100dB and a minimum of -60dB, which produced the following plot:

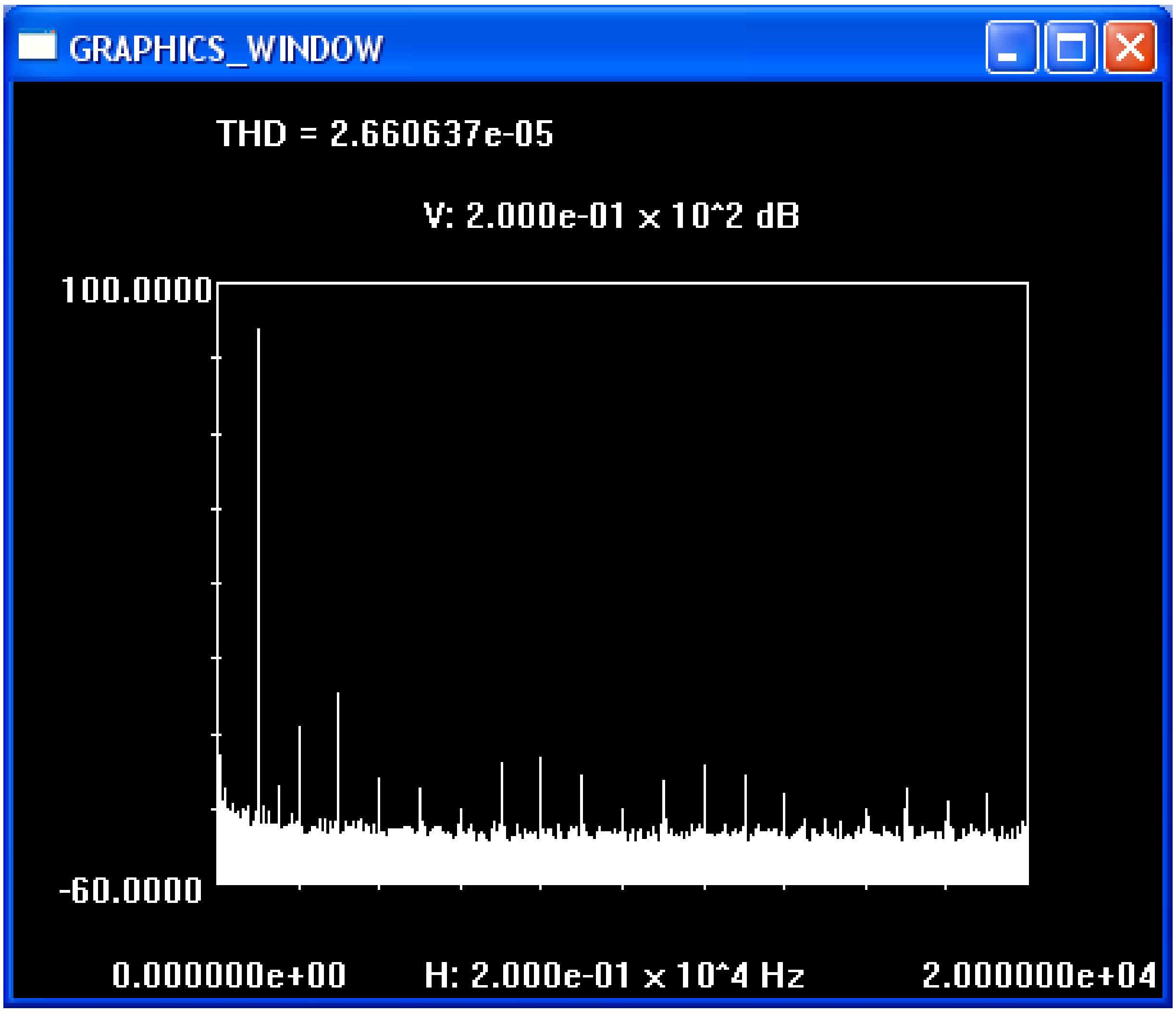


Figure 39: Frequency response magnitude and THD+N of 1kHz sine wave loopback test.

Note that because we enabled the T Option in the FFT Menu, the THD+N is calculated and displayed on the top of the graph in Fig. 39. Note also that the THD+N calculated according to Eq. 5 is not expressed in %, but is rather a magnitude ratio. This value represents the fundamental precision to which the Virtual Crossover can carry out THD+N measurements, and it also indicates the distortion level to be expected in music or test waveforms processed by the Virtual Crossover.

21 Measurement Examples – Impedance Measurement

In this section I measure the input impedance of an electrostatic speaker panel that I constructed based on the articles by Roger Sanders (*Speaker Builder*, 1980 and 1990).

The first step was to program a "blank" crossover into Virtual Crossover SDRAM – please see the MLS measurement example for a discussion of how to do this. Next, selecting M from the Main Menu to set up a measurement, then M again to set up an MLS sequence, I set up a sequence of order 12 and set num_meas_points to 4096. I specified 0 additional delay. I also enabled Raw Mode to monitor the amplitude on waveform channel 1 of the Virtual Crossover. Going next to the FFT Menu, I toggled Impedance Mode on by using the I option, and I entered a value of 9.77Ω for the series resistance.

I connected the 1L output of the Virtual Crossover to the input of my power amplifier. At the output of the power amplifier I put a 9.77Ω resistor in series with the electrostatic panel input. (The electrostatic panel is driven by a step-up transformer, so the resistor was connected in series with the transformer).

I connected the Left Input of the Virtual Crossover to the output of the power amplifier, and the Right Input of the Virtual Crossover to the point where the series resistor connects to the step-up transformer.

Going back to the Main Menu, I set the DAC amplitude very low, to a value of 1 initially, in order to bring the amplitude up slowly. I then started the MLS measurements by typing G from the Main Menu. I slowly brought up the amplitude to about 20, which did not show any sign of clipping the Virtual Crossover input (which occurs at ±1). (You can adjust the amplitude in a couple different ways. One is to press the up- and down-arrow keys while the measurement is in progress. The other is to temporarily stop the measurement by typing Q, then using the V option (or arrow keys) to set the volume, and start the measurement again by typing G. It is often faster to temporarily stop the measurement, because you are limited how fast you can change the volume while the measurement is in progress). Since RAW Mode was enabled, I saw the following display:

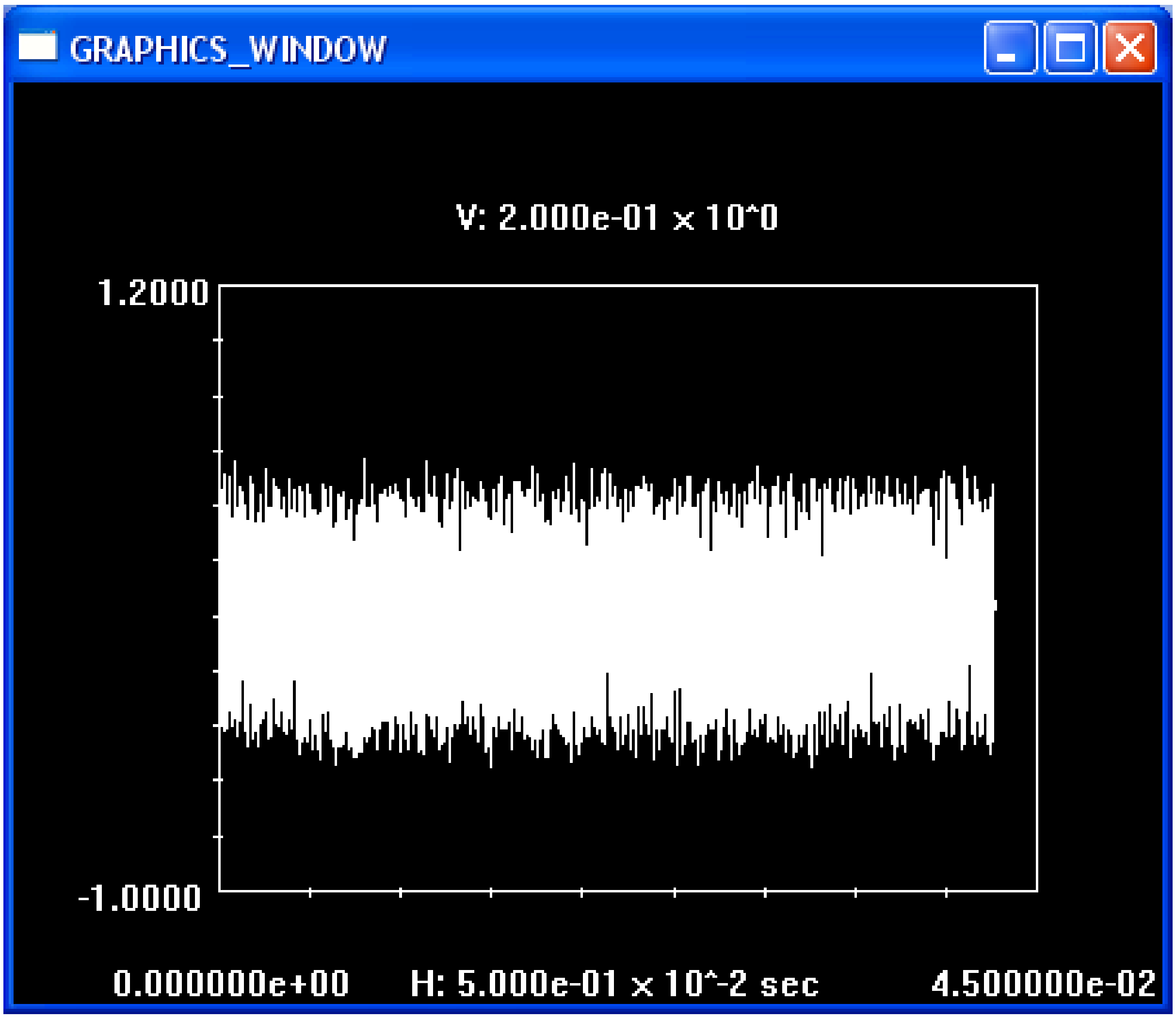


Figure 40: Waveform1 in Raw Mode during ESL impedance measurement.

Once the amplitude was set, I toggled off Raw Mode and re-started the measurement. Since we are in Impedance Mode, two channels are captured during each measurement. For monitoring of amplitude it is best to look at waveform 1 (Left Input), since this signal is directly at the output of the power amplifier and is usually larger than waveform 2. Once Raw Mode is switched off, the plot shows the impulse response of waveform 1:

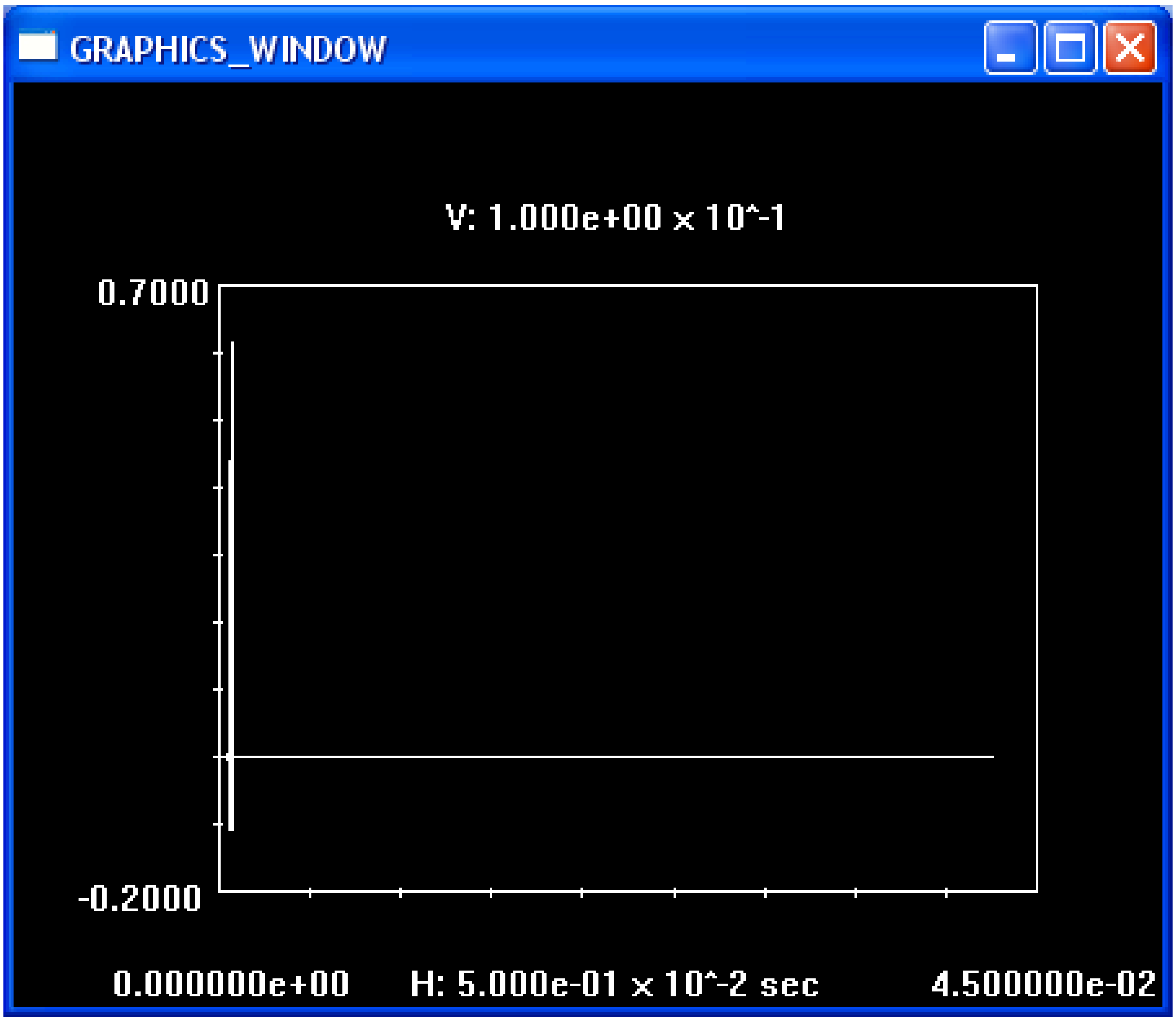


Figure 41: Impulse response of waveform 1 during ESL impedance measurement.

After the measurement is stopped by typing Q, you can also plot the impulse response of waveform 2 using the Time Waveforms Menu P option:

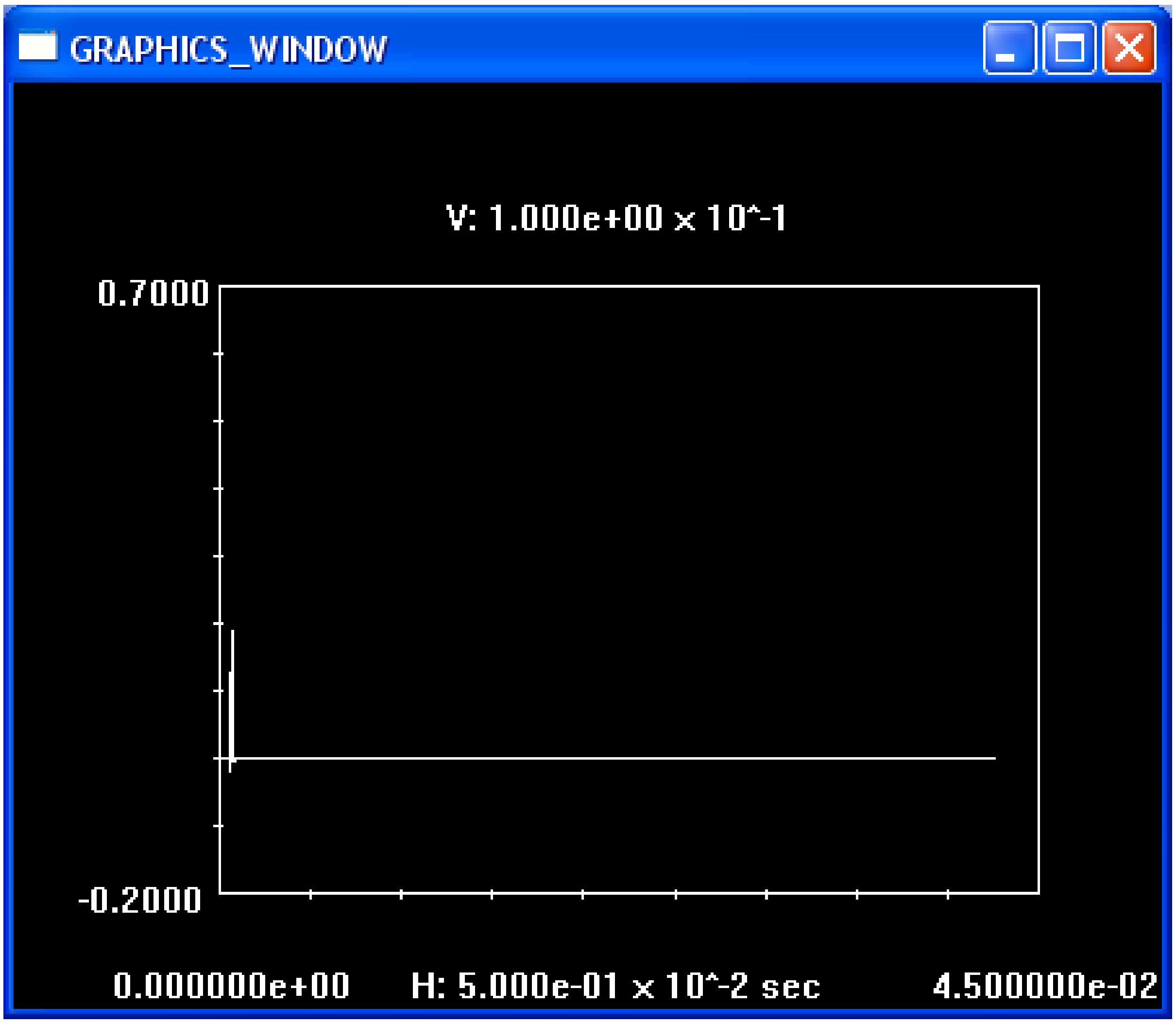


Figure 42: Impulse response of waveform 2 during ESL impedance measurement.

To determine the input impedance, go to the FFT Menu, type B to take the FFTs of both waveforms 1 and 2, then D to form the ratio of the FFTs appropriate for impedance. Since Impedance Mode is enabled, frequency response magnitude plots will be in units of Ohms. You can use the P option to set the maximum and minimum value for the Ohms vertical scale – I chose 20Ω and 0Ω as the axis limits. Also, using Cursors Mode I limited the frequency response magnitude plot to a maximum frequency of 25kHz. To plot the impedance magnitude, choose the A option (amplitude plot) from the FFT Menu and type 1 to select FFT 1. The following shows the input impedance magnitude thus obtained for the electrostatic panel:

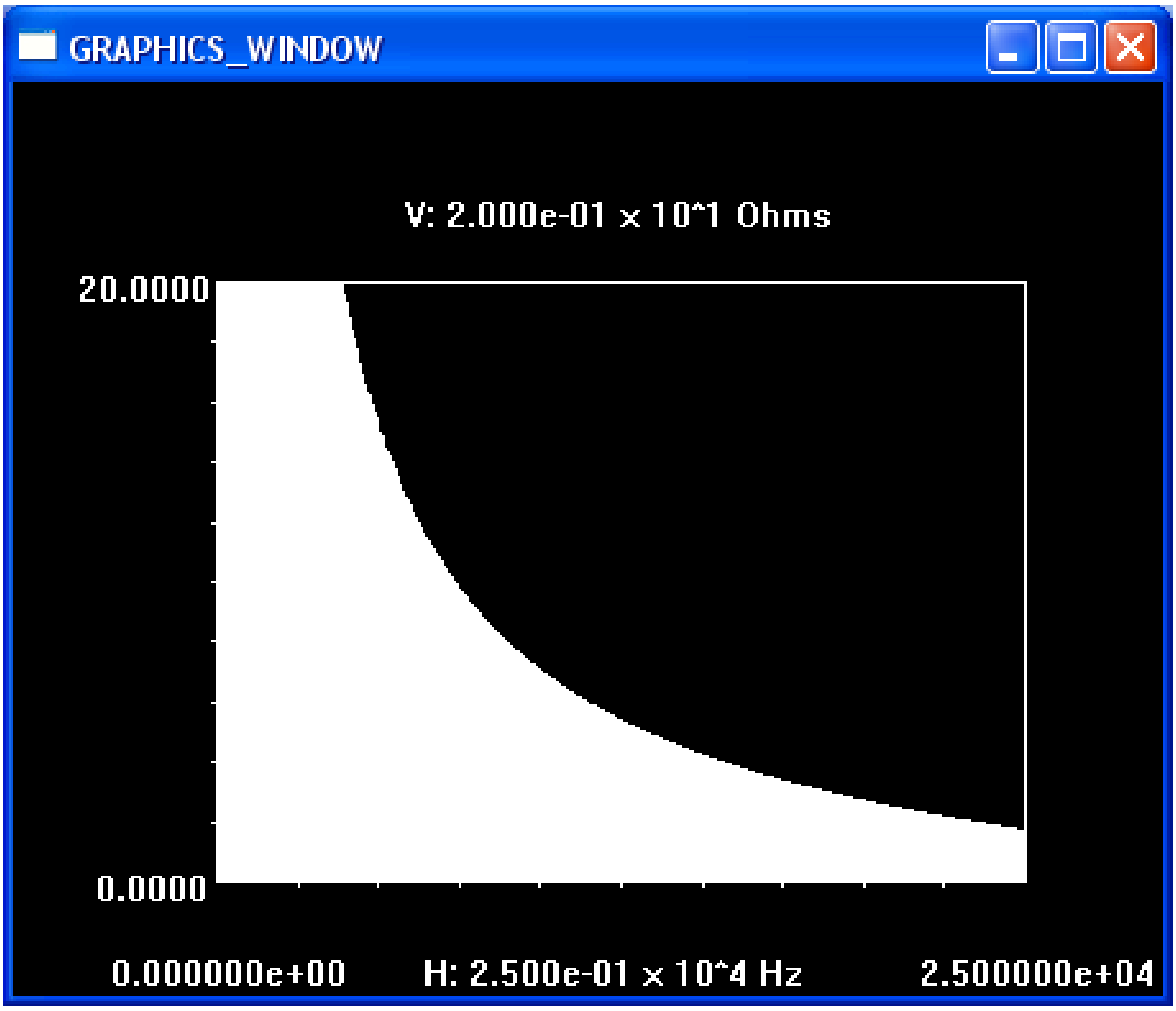


Figure 43: Input impedance magnitude of electrostatic panel.

As can be seen from Fig. 43, the electrostatic panel behaves like a capacitor, exhibiting a decreasing impedance magnitude with frequency. At 20kHz the input impedance magnitude is down to about 2.78Ω. It is clear from Fig. 43 that electrostatics can present a demanding load to power amplifiers at higher audio frequencies.

We can also plot the phase angle of the electrostatic input impedance by choosing the H option in the FFT menu and typing 1 to select FFT 1. Cursors Mode can also be used to limit the frequency extent of the plot, in this case to 25kHz:

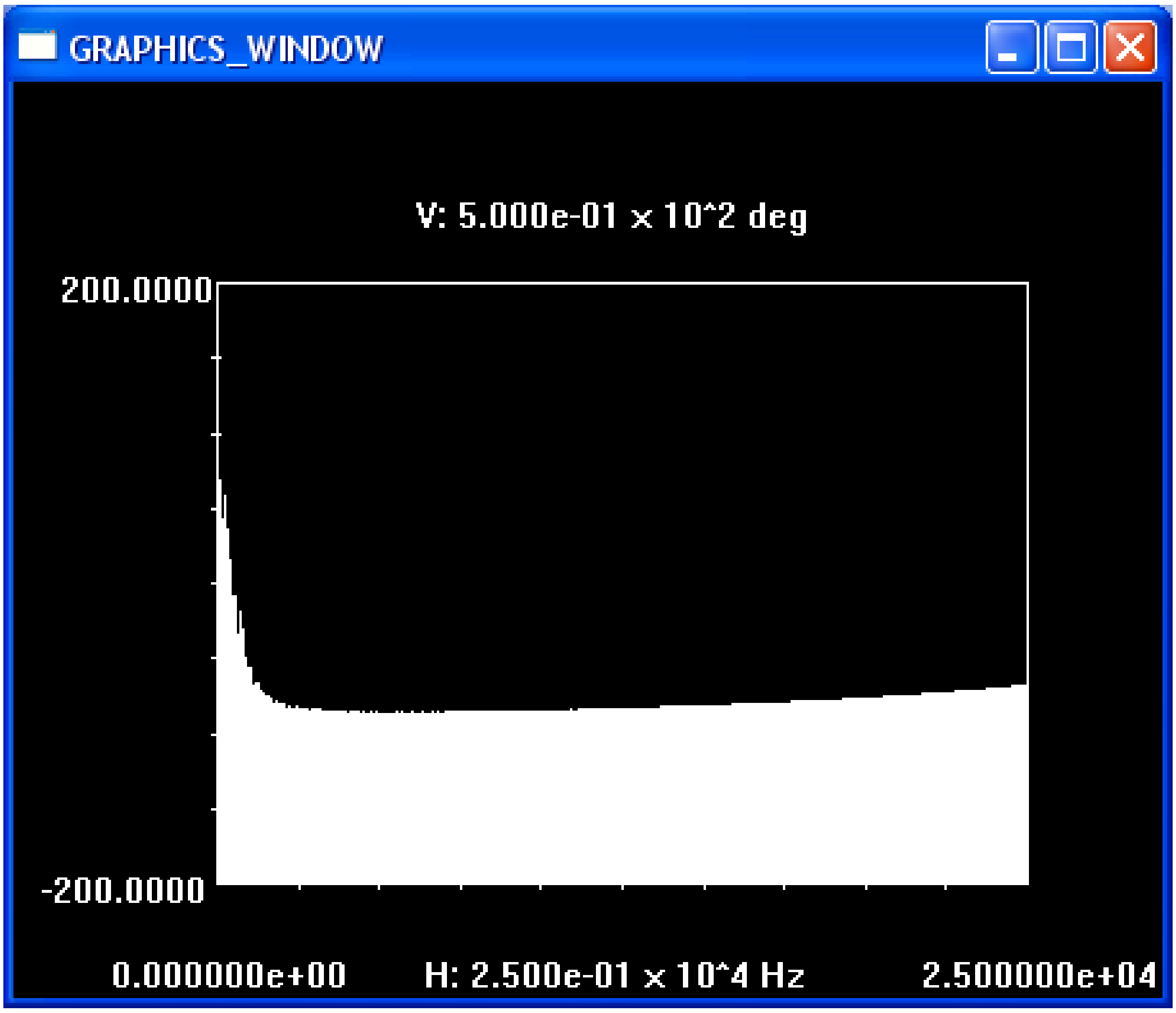


Figure 44: Phase angle of electrostatic panel input impedance.

It is seen from Fig. 44 that the phase angle nearly reaches the −90° value that would be expected for a capacitor, but then increases with frequency indicating a resistive component in the impedance. Also note that there is evidence of a resonance at lower frequencies, either with the step-up transformer or due to the resonance of the electrostatic driver itself.

22 The vc.in file

When you run the vc.exe program, there must be a file called vc.in of the proper format in the same directory, or vc.exe will not execute. This section presents a sample vc.in file along with a discussion of the options available in the file. You must strictly adhere to the format of the vc.in file; each value is preceeded by one or more lines of comments, followed by the value itself. A comment line is identified with a # in the first column.

Note that when a value is read from a line in vc.in, vc.exe takes the first value on the line. So if you are trying different values for a parameter and you want to remember which values you have already tried, you can leave them on the line after the current value you are trying, as long as they are separated by at least one space.

Also note that some of the parameters in vc.in were used during the development and debugging stages for the Virtual Crossover, and since it is not expected that the user will want to use these features, a detailed explanation is not given.

In the following listing, each line begins with n:, where n is an integer; these numbers do not appear in the actual file, they are placed in this listing in order to be able to easily refer to each line in the discussion:

```
1:# vc_attached (0 if not, 1 if vc is attached) =
2:0
3:# VendorID =
4:
5:#ProductID =
6:
7:#max_usb_bytes_per_message (for 2 endpoints) =
8:128
9:#timer_mode (0=send once, 1=continuous send) =
10:0
11:#timer_num_sec_delay (delay between sends if timer_mode=1) =
12:0.2
13:#test_measurement_delay_between_waveforms (in seconds) =
14:0.0
15:#delt in seconds =
16:1.041666666666666e-5 6.5104167e-7 2.083333333333e-5
17:#num_meas_points =
18:65536 4096 32768 8192
19:# steps in playback gain in dB (playback_gain_step_db):
20:1.0
21:#number of fft_size's in capture and playback buffer sizes:
22:4
23:#shift (lshift instruction) of max_fft_size to determine playback buffer old pointer delay:
24:1
25:#filter truncation factor =
26:1e-5
27:#
28:# Circuit elements for A/D input loading in impedance measurements:
29:#
30:# RL1 (e.g. R17 or R20 in VC schematic) =
31:1e4
32:# CL1 (e.g. C5——C27 or C6——C31 in VC schematic) =
33:44e-6
34:# RL2 (e.g. R5 or R8 in VC schematic) =
35:4.64e3
36:# CL2 (e.g. C13 or C18 in VC schematic) =
37:100e-12
38:# CL3 (e.g. C14——C19 in VC schematic) =
39:200e-12
40:#
41:# MLS parameters:
42:#
43:#r_mls (delault number of mls cycles in measurement - should be ¿=3 ) =
44:4
45:#mls_max (maximum mls normalization for D/A waveform - ¡=0 to disable) =
46:0.9
47:#mls_rslt_fact (multiplies mls result) =
48:2.5
49:#display_mls_raw (option to display raw mls data without correlation step) =
50:0
51:#
52:# End MLS parameters
53:#
54:# Color parameters
55:#
56:# primary colors:
57:#255 255 255 white
58:#0 0 255 blue
59:#0 255 0 green
60:#255 0 0 red
61:#255 255 0 yellow
62:#255 0 255 magenta
63:#0 255 255 cyan
64:#
65:# foreground color:
66:255 255 255 white
67:# background color:
68:0 0 0 black
69:#
70:# plot view extents parameters
71:# minx_div, minx_mult =
72:8 1
73:#
74:# maxx_div, maxx_mult =
75:8 7
76:#
77:# note: miny is at bottom of screen, it is actually larger than maxy
78:# miny_div, miny_mult =
79:8 7
80:# maxy_div, maxy_mult =
81:8 1
82:# end plot view extents parameters
83:#
84:# number of playback channels (put zero if there are none) =
85:6
86:# filter index for playback channel index 0 (negative fft_size for no filter) =
87:-8192 0
88:# filter index for playback channel index 1 (negative fft_size for no filter) =
89:-8192 0
90:# filter index for playback channel index 2 (negative fft_size for no filter) =
91:-8192 1
92:# filter index for playback channel index 3 (negative fft_size for no filter) =
93:-8192 1
94:# filter index for playback channel index 4 (negative fft_size for no filter) =
95:-8192 2
96:# filter index for playback channel index 5 (negative fft_size for no filter) =
97:-8192 2
```

```
98:# capture buffer index for playback channel index 0 =
99:0
100:# capture buffer index for playback channel index 1 =
101:0 1
102:# capture buffer index for playback channel index 2 =
103:0
104:# capture buffer index for playback channel index 3 =
105:0 1
106:# capture buffer index for playback channel index 4 =
107:0
108:# capture buffer index for playback channel index 5 =
109:0 1
```

Line 2: As mentioned elsewhere in this manual, vc.exe can be run in two different modes, one where the Virtual Crossover is attached to the PC over the usb port, and one where it is not. The integer that you enter in this line, either 0 or 1, tells vc.exe whether or not it should look for the Virtual Crossover.

Lines 3-6: The Product ID and Vendor ID are used to identify the Virtual Crossover on the usb bus. These values should not be changed (unless the firmware in the Virtual Crossover is changed), otherwise vc.exe will not be able to find the Virtual Crossover on the usb bus. The numerical values have not been included in the listing because they are still in the process of being finalized.

Line 8: max_usb_bytes_per_message is the maximum number of bytes that can be sent or received in one usb message. This value is dictated by the usb firmware and should not be changed, unless the firmware is changed. (Well you could make it smaller if you wanted, but then usb communications would be slower).

Line 14: If you set up a measurement with a non-periodic waveform, test_measurement_delay_between_waveforms is an additional delay between applications of the waveform. However, for large numbers of measurement points, there is also a consdierable delay due to transfer of the measurement results over the usb port; this delay usually provides sufficient delay between waveform outputs.

Line 16: delt is the audio sample period in seconds. It is important to set this to the value corresponding to the sampling rate you have chosen for the Virtual Crossover.

Line 18: num_meas_points is the number of points desired for measurement results. It must be a power of 2 so that FFTs can be used. It is possible to change this value after the program starts up, for example when you set up an MLS measurement the program asks you if you want to change num_meas_points.

Line 20: This is the step in dB that the output gain will take when you press the up- or down-arrow keys to change the volume.

Line 22: This parameter controls the size of the capture and playback buffers. You need more than one fft_size in order to keep up with the demands of streaming audio. 4 fft_sizes has worked well for me so far.

Line 24: This parameter determines how many fft_sizes the playback buffer old pointers are delayed. This delay is to ensure that the Virtual Crossover has enough time to calculate new output data before it is needed. This value must be an integer. If you start getting underrun errors you can try making this value larger. Usually a value of 1 or 2 works well.

Line 26: The filter truncation factor is used to limit the size of crossover filters in the time domain; values below this factor times the waveform maximum value are treated as zero. It is important to limit the number of non-zero points, because that is the number of points that must be discarded at each FFT calculation when processing audio signals through the filters. Selecting this parameter is a tradeoff between accuracy and computational speed.

Lines 27-39: When the Virtual Crossover is used to measure impedance, it is important to realize that the input A/D circuit can alter the impedance of the device being measured; this is because the input impedance of the A/D circuit is not infinite and it is a function of frequency. This loading should not become significant until the impedance of the device under test reaches the order of kilo-Ohms. Since we know what the A/D input circuit is, we can compensate for its effect on the measured impedance (except that the tolerance of the components in the A/D circuit will vary, so the compensation will not be perfect). The elements in lines 27-39 are the element values in the A/D circuit used to compensate for this effect. These should not be changed unless you know that the A/D circuit is for some reason different from that described by these component values.

Line 44: When you carry out an MLS measurement, the Virtual Crossover outputs a number of MLS sequences as specified by this parameter for each measurement; it then captures the next-to-last cycle as the measurement result. You can control the number of MLS sequences per measurement by changing this parameter.

Line 46: When an MLS sequence is set up by the vc.exe program, the sequence values are multiplied by the mls_max value before being sent to the Virtual Crossover's memory. This parameter can be used to help avoid clipping and its resulting distortion in MLS measurements.

Line 48: After the vc.exe program has obtained the results of an MLS measurement and has carried out the cross-correlation to obtain the impulse response, the result is multiplied by this parameter. It is basically used to normalize the impulse response to a desired magnitude range. (For example if your time-domain plots have $\pm$ 1 as the axis limits, you can set this parameter so that the impulse response is readily visible on the plots).

Line 50: This is the default setting for the "raw" option in MLS measurements. The raw option allows you to see the raw MLS data, before cross-correlation is carried out to obtain the impulse response.

Lines 66,68: These lines allow you to set the foreground and background colors for plots, as triplets of RGB values in decimal. (This feature is not yet available in the Linux version of the vc program).

Lines 70-82: These parameters allow you to set the relative size of the screen plots. This is done through a series of _div and _mul values. Basically, vc.exe comes up with a maximum window size, and then uses these values to determine the plot extents. For example, to determine the first parameter minx, the left-most extent of the plot along the horizontal direction, it would first divide the horizontal window size by 8, and then multiply that result by 1, to get the starting x position. Similar calculations are carried out for the other 3 parameters specifying the extents of the plots. (This option is not yet available in the Linux version of the vc program).

Line 85: Set this parameter to the number of playback channels that you require in your system. It doesn't mean that you always have to use all the channels – the purpose of this parameter is to avoid processing of playback channels that are never used.

Lines 86-97: Each playback channel has a filter index associated with it (the values in the vc.in file initialize the filter indices, but you can change them later using the L option from the Main Menu). There are basically two different ways of setting the filter indices. If you are actually going to have crossover filters, then you want to associate a playback channel with the crossover filter that you want to process the output to that playback channel. For example, if playback channel 0 goes to the bass driver, then you want to assign the bass crossover filter to playback channel 0. The second integer in each line represents this kind of assignment.

On the other hand if you typically are not going to set up crossover filters or if you just want to have the Virtual Crossover come up in "pass-through" mode, where the output is the same as the input, then you can assign the filter indices to minus the FFT size, as is done with the first integers in each assignment line.

Lines 98-109: Each playback channel has a capture channel associated with it; this is where the input data is obtained that will be processed for the playback channel. Since the input is always stereo audio, there are only two capture channels, denoted by integers 0 (left) and 1 (right).