Chapter 9. Filter and Crossover Design

Background

Multi-way speaker systems can provide the best possible acoustic performance with each loudspeaker specially designed to cover a limited section of the frequency spectrum. To obtain this performance one must ensure that the electrical signal input to the different loudspeakers is divided in such a way that each speaker receives only the signals it is required to reproduce.

The Crossover/CAD tools have been provided to enable a thorough examination and design of the **FIR, passive and active parallel, series and cascaded filters/crossovers**. The parallel filter design enables the designer to treat each driver independently. This is possible because the driving amplifier is usually a voltage source. Only one driver is taken into account when each optimisation is performed. Crossover design is a difficult task and this manual is not intended to substitute a text book on crossover design. The user is well advised to seek additional information on the subject and the list of texts in the Appendix would be a good starting point. Please load now the testXXXX.wfr data file. Two approaches could be taken when working with the crossover tools. First would assume, that you design "less then typical" crossover. It could be a band-pass section which has +18/-6 dB slopes or 5-th order high-pass filter. You would then be required to create schematic using the technique described in Chapter 8 and perform your own calculations for the component values. After you entered the components' values, SoundEasy will then help you with the analysis of your customised crossover.

Channel Filters or Crossovers

The 'Crossover' tool enables you to design and analyse:

1.   Complete crossover network OR

First of all, design of a complete crossover on one screen is convenient. You can see the complete design and understand better the interaction between the individual electronic components and channels. Secondly, if your intention is to use 'System' screen for further analysis of the system, you need to use this approach. This is because **the 'System' screen is designed to analyse complete crossovers rather than individual filters**. You may stop at this moment to think about how to plot input impedance of individual channel when the complete crossover is drawn on the screen ?. We suggest, that in many cases, you will be able to change a component value in your crossover to make it act as a "switch" allowing you to keep the schematic configuration intact, but a channel disconnected. For example: two-way, second order crossover has the HP filter capacitor (C4) reduced to a VERY small value, so it practically cuts-off the HP section from analysis. Input impedance of the crossover now only includes the LP section - see Fig 9.1. If your crossover topology prohibits you from using this approach, you may need to include extra "switching" components in your crossover.

Please refer to section "First Cut Design" for more information about complete crossover design approach.

2.   Individual channel (woofer, midrange, tweeter…) filter

Individual filters are useful for two reasons: (1) allow you to easily analyse input impedance of a parallel crossover and (2) driver files with the filter or EQ circuit can be loaded back into the 'Box' module and used in conjunction with the enclosure frequency responses. This is the approach you would take to design **an active subwoofer with EQ circuit**. Both ways will include driver frequency response and input impedance in modeling all transfer functions of your filter or crossover. Also, both approaches can be used with filter/crossover optimizer.

Using Built-In Filters

Five types of filters are built-in for faster network implementation. These are: Butterworth, Bullock, Bessel, Linkwitz and All-Pass filters. The filters are edited and selected by "Built-in Crossover" under "Edit" main menu option. This will open a dialog box from which the user may set-up the required configuration. Most of the filters are available in 1,2,3,4,5 and 6-pole versions. Low, band (symmetrical) and high-pass versions are available as passive and active networks (see Fig 9.2).
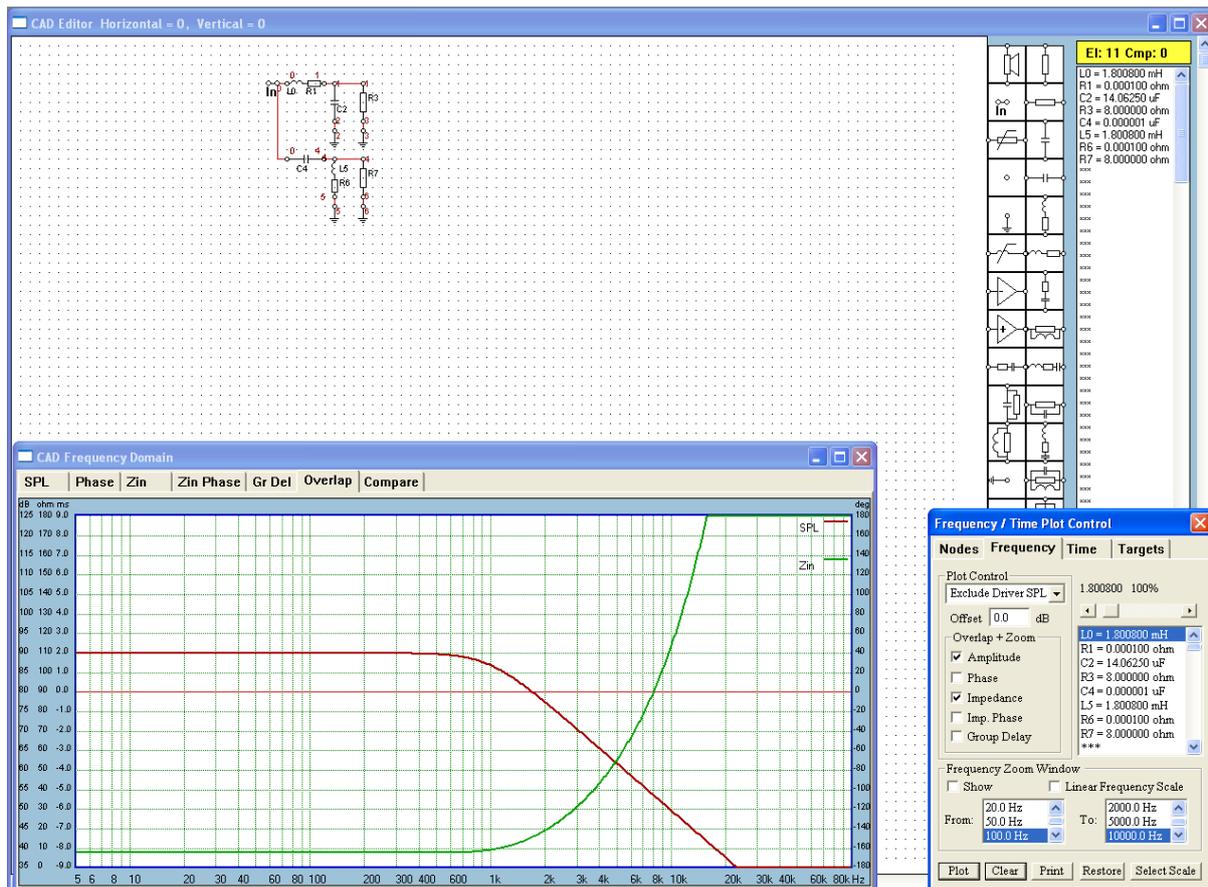
Fig 9.1 the HP filter capacitor (C4) reduced to a VERY small value

Example: In order to select the Butterworth band-pass crossover, Select "Built-in Filter" from the "Edit" menu and when a dialog box appears, follow the steps described bellow.

1. Select the required Filter Configuration from the "1. Filter Configuration" list box.
2. Select the filter (eg: Butterworth) type from the "2. Filter Type" list box.
3. Edit the high-pass or low-pass cut-off frequencies, the "-3dB" frequencies in
    "3. Filter Limits" data entry fields.
4. Enter required load resistance R(load) under "4. Passive Network" or arbitrary Ro or Co for "4. Active Network".
5. Press "5. Done" button

You can now obtain a hard copy of the selected values by pressing the 'Print' button. Otherwise, click-on "Done" button to continue with the design process. SoundEasy will remove the dialog box and create the required filter. When the process is finished, a complete crossover is drawn in the screen together with the parts list displayed in the list box on the right. This filter is loaded by a single resistor. Normally, the user would want to evaluate the circuit's performance when the filter is loaded by a loudspeaker (and possibly a compensation network). To simulate this configuration for real driver, you must **load the data file first** (refer to Chapter 2), then enter required filter frequencies, select slope orders and type using previously described dialog box. There is no need to edit the load resistance. The 'Crossover' tool is now informed that the driver data file is loaded and the DC resistance of the driver is automatically selected for the calculations as the load resistance. Click on the 'Done' button when finished. When the network appears on the screen, replace the load resistor with the loudspeaker symbol - see Fig 9.4. In this way you tell the program that you wish to use data calculated by the previous tool, the 'Box' module.

Replacing the resistor is performed in the same way as used when replacing a wrong component. If in doubt, please refer to Chapter 8. Fig 9.3 shows an example data file loaded and some graphs plotted using the "Frequency Domain" window**. "Clear CAD"** menu option clears schematic only and **"Clear Data"** menu option clears all data and the CAD screen.

9.2
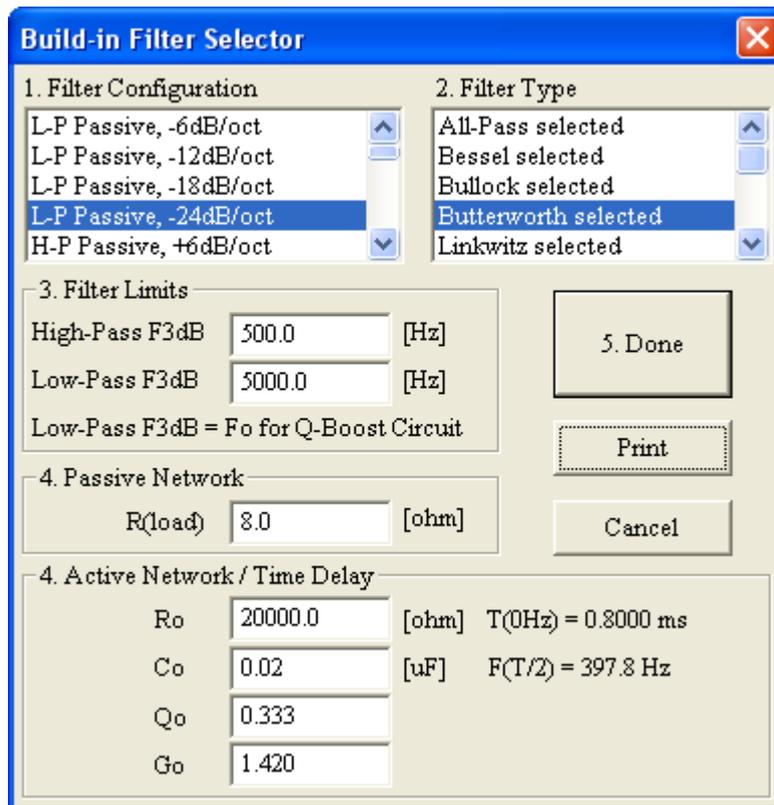
Fig 9.2 Built-in Filter Selector dialogue box
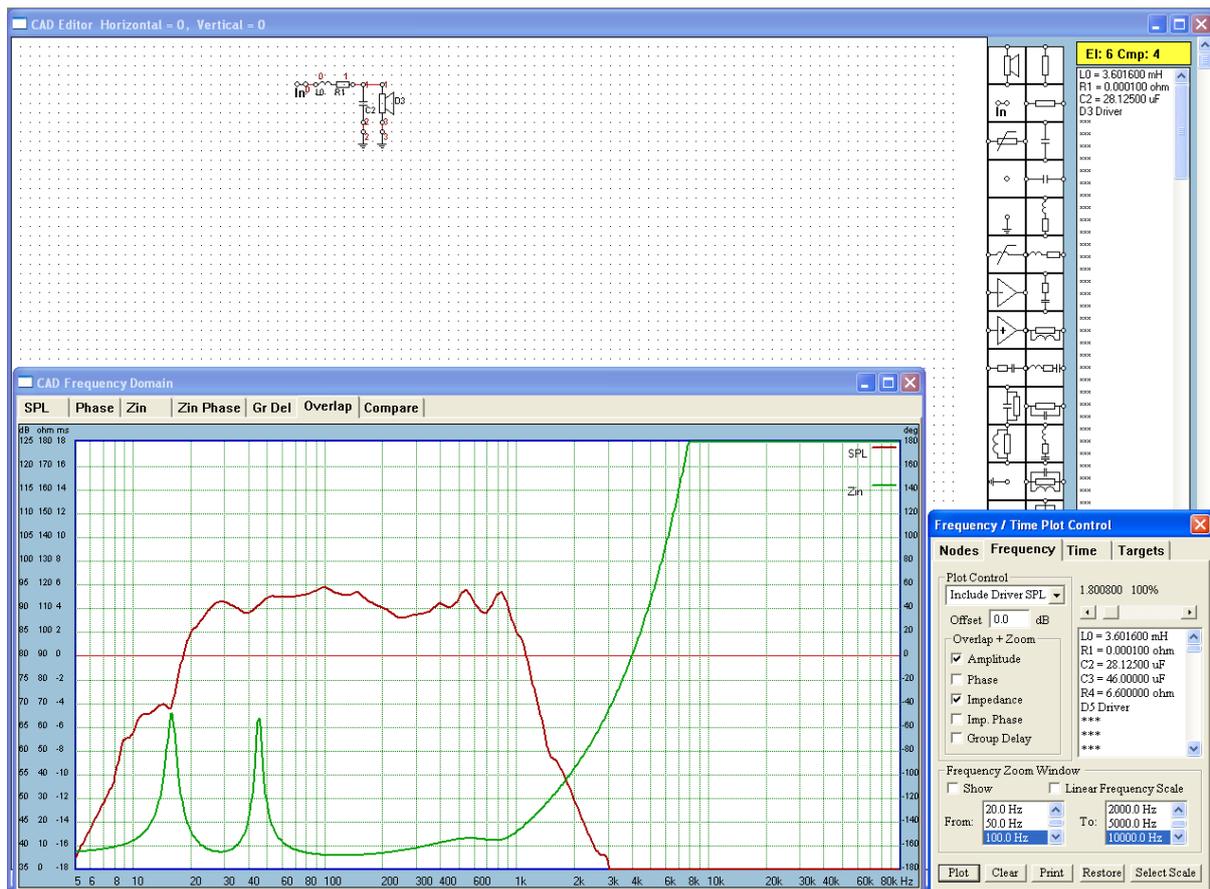
Fig 9.3 Driver file was loaded first and then built-in filter was selected

Using Built-In Crossovers

        Three types of crossovers are built-in for faster network implementation. These are: Butterworth, Bessel and Linkwitz crossovers. The crossovers are edited and selected by "Built-in Crossover" under "Edit" main menu option. This will open a dialog box from which the user may set-up the required configuration. Most of the crossovers are available in 1,2,3 and 4-pole versions, 2-way to 5-way configurations. Low, band (symmetrical) and high-pass versions are available as passive and active networks (see Fig 9.4).

Example: In order to select the Butterworth band-pass crossover, Select "Built-in Crossover" from the "Edit" menu and when a dialog box appears, follow the steps described bellow.

1. Select the required Filter Configuration from the "1. Crossover Configuration" list box.
2. Select the filter (eg: Butterworth) type from the "2. Crossover Type" list box.
3. Edit the high-pass or low-pass cut-off frequencies, the "-3dB" frequencies in "3. Crossover Limits" data entry fields. NOTE: The first crossover frequency is considered to be the one between the lowest frequency driver (say, woofer) and the immediate next higher frequency driver (say, upper bass) and so on….**You must enter the frequencies in increasing order, even if you do not use all data fields**. For example, if you only intend to design a two-way system with a crossover frequency of 1000Hz, enter 1000Hz for the "First F3dB, then enter a dummy data of 2000Hz for the Second F3dB, then again a dummy data of 3000Hz for the third F3dB and so on..
4. Enter required load resistance RL 1- RL 5 for individual load resistance for each filter, (or arbitrary Ro or Co for "4. Active Network"). **Please note, that individual load resistance option is implemented for the most popular filters: Butterworth and Linkwitz. For other filter types, please enter R(load) value.**
5. Press "5. Done" button

Fig 9.4 Built-in Crossover Selector dialogue box

Active Crossover

Note: Successful application of active networks must consider hardware issues like biasing, supply voltage, driving point impedances, buffering, clipping, feedback, distortion and much more. This program is only concerned with transfer functions of the circuits being modelled.

Filtering circuits employing operational amplifiers have been extensively debated and described in details in professional literature. The task of this module is therefore, to provide the user with an easy to use and flexible tool, capable of analysing active network behaviour. CAD system operation is exactly the same as for passive networks however, model used for inverting and non-inverting amplifiers requires more detailed presentation.

Operational Amplifier

In order to accommodate practical operational amplifiers, the program uses VCVS (Voltage Controlled Voltage Source) to model the operational amplifier. Practical amplifiers have finite gain, finite input impedance and greater than zero output impedance. All those elements were incorporated in the VCVS model so it become a non-ideal VCVS. Default values for the low frequency model of :

1. Inverting amplifier are: Rin=10Mohm, Rout=100ohm and Ku=100000. These parameters are taken directly from data sheets of the product.

2. Non-inverting amplifier are: Rin=1000Mohm, Rout=0.001ohm and Ku=1.0. These parameters are pre-calculated as shown in the example below. Please note, that ALL parameters are entered as positive numbers.
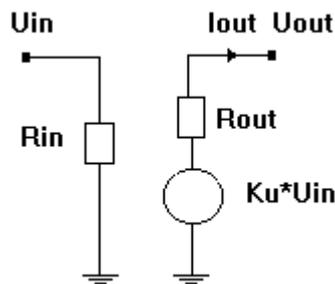

Fig 9.5 Single-input OPAMP model

All components of the model can be edited from dialogue boxes. Fig 9.5 shows model used for representing the two types of amplifiers. When the inverting amplifier is used, Ku denotes **open loop** gain. The "in circuit", actual input and output impedances can be calculated observing Miller's theorem. When the non-inverting amplifier is used, the default values have been set to represent a good quality voltage follower and Ku represents **closed loop** gain. In general case, the closed loop gain of a non-inverting amplifier is set by two external resistors, (feedback resistor - connected between output of the amplifier and the inverting input and another resistor - connected between the inverting input and the ground) so that, for example, a non-inverting amplifier with a gain of 100 can be constructed. You can simulate this case here by setting 'Gain' to 100 in the dialogue box and when you actually build the circuit, you will use two additional resistors to set the gain, as described above. Other parameters should be calculated as in the following example:

Example: Operational amplifier you selected for non-inverting application has the following data as provided by manufacturer: Zin=1Mohm, Zout=100ohm and open loop gain of Aol=10000. Your application calls for an amplification in order of 20. Set "Gain" from the dialogue box to 20 - remember, it is the closed loop gain that is required here, and when you finally build your circuit, you should use external resistive feedback to set the gain.

Rin(non-inverting OPAMP) = (1+Aol/Gain)*Zin = (1+10000/20)*1 = 51Mohm
Rout(non-inverting OPAMP) = Zout/(1+Aol/Gain) = 100/(1+10000/20) = 2ohm.

The following data should be entered into the non-inverting operational amplifier dialogue box: Rin=51Mohm, Rout=2ohm and Gain=20. In case you would use the same hardware as an inverting amplifier, you should enter the original data: Rin=1Mohm, Rout=100ohm and Gain=10000. We recommend, that you use load resistor from output of the amplifier to ground to simulate real life loads. Modification of the default values, provides the user with a powerful mechanism to customise behaviour of both amplifiers. Should your application require an implementation of a differential amplifier, use the following configuration (use the following data):

1. A1: Rin1=10Mohm, Rout1=100ohm, Gain1=1.
2. A2: Rin2=10Mohm, Rout2=100ohm, Gain2=1.
3. A3: Rin3=10Mohm, Rout3=100ohm, Gain3=100000.
4. A1 input is the inverting input and A2 input is the non-inverting input.

This configuration is used in active delay network of the first order shown in previous chapter. It is observable, that A1 acts as a simple voltage inverter and A2 is a unity gain buffer. Existence of greater than zero output impedances in A1 and A2 (100ohm) allows us to connect their two outputs in parallel - see Fig 9.6. As with the passive delay networks, one can observe flat amplitude response.
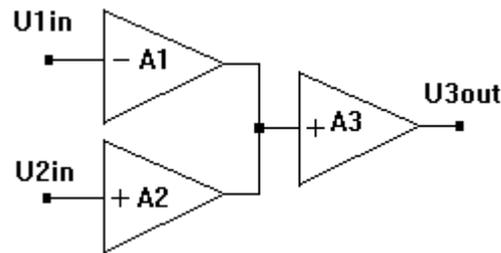

Fig 9.6 Example of a Differential OPAMP built from 3 single-input OPAMPS

When analysing active circuits, three plots will be of importance to the designer: (1) amplitude response, (2) phase response and (3) group delay. Input impedance of the circuit was of prime importance for passive compensation/crossover networks, as it determined quality of the crossover load. In case of active networks, loudspeakers are isolated from the active crossovers by the power amplifiers. Also, buffer preceding the active crossover will always provide near zero output impedance eliminating the need to examine the loads presented to it by the active crossover.

Network Example

Frequently recommended active crossover is a third order Butterworth filter. It will exhibit maximally flat magnitude response, it has sharp cutoff characteristics of -18dB/oct and it has flat voltage and power frequency responses with a gradual change in phase across the band. Please refer to Fig 9.8 and 9.9 Several tropologies have been proposed for this filter, out of these, multiple feedback approach offers good tradeoffs in circuit complexity, component spread and sensitivities.

A common design approach would be to design the active filter with unity gain and adjust system gains by tweaking power amplifiers' gains. We also strongly **recommend that the final gain of the active crossover filters is equal to 1, as it enables proper functioning of the optimizer modules**. Since loudspeakers are connected directly to power amplifiers, there is also no need to provide impedance compensation for the drivers. Power amplifiers will provide very good level of dumping. Separating the spectrum before the final power amplifiers makes it easier to control undesirable effects of clipping high frequency signals in the presence of high level low frequency signals.

When modelling active crossovers using SoundEasy, the loudspeaker can be connected directly to the output of the operational amplifier. Theoretically it is possible, (remember, we are not concerned with the practical hardware limitations) since the negative feedback of this particular configuration reduces output impedance of the amplifier to a small fraction of an ohm, and the amplifier starts to behave like an almost ideal voltage source.

Additionally, you can reduce output resistance of the amplifier (Rout) to a small value. Most of the "feedback" type of arrangements would fell into this category. Because of the above, you can still use the same techniques of "including or excluding" driver as it was explained for the passive networks, and the same substitute of loudspeaker symbol for the load resistor to include frequency/phase/group delay and impedance responses of the driver in the active network analysis.

9.6
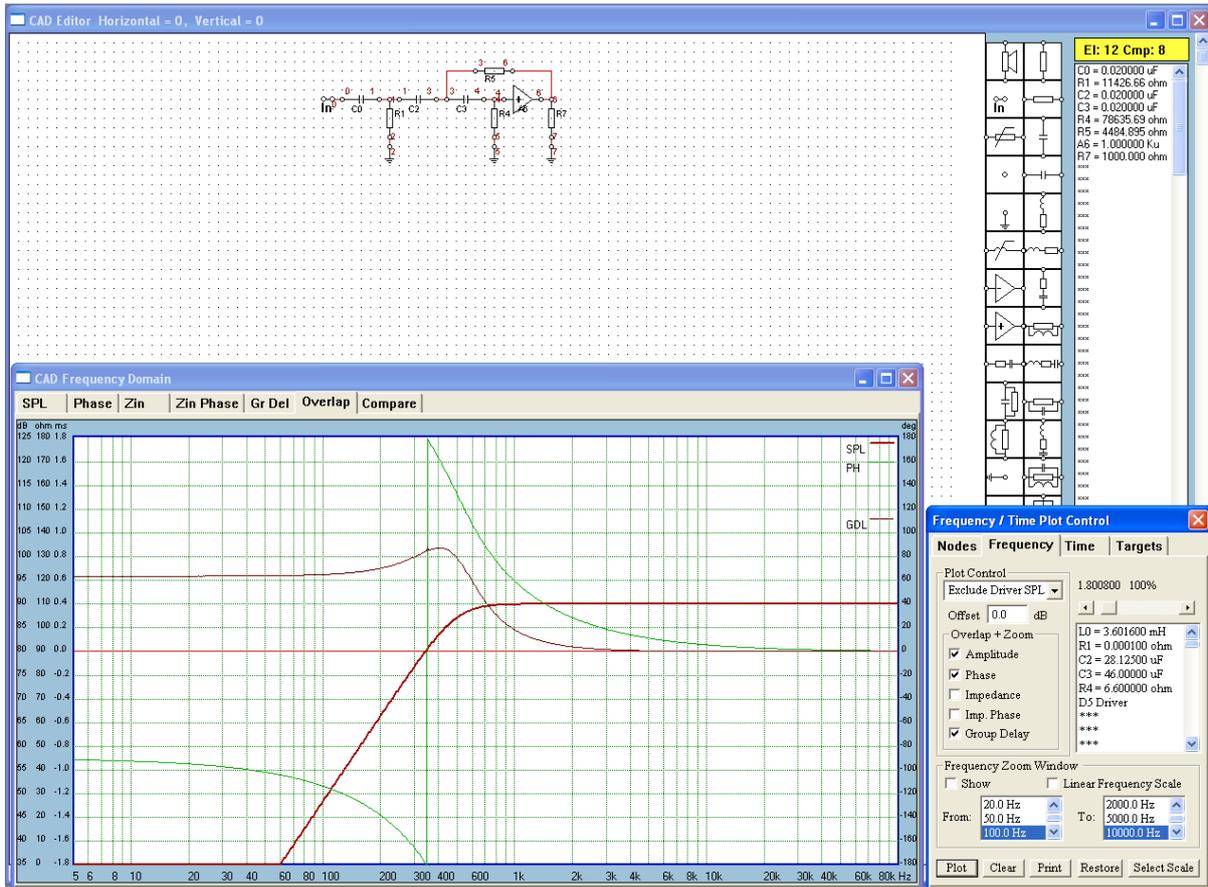
Fig 9.8 Butterworth +18dB/oct HP filter.
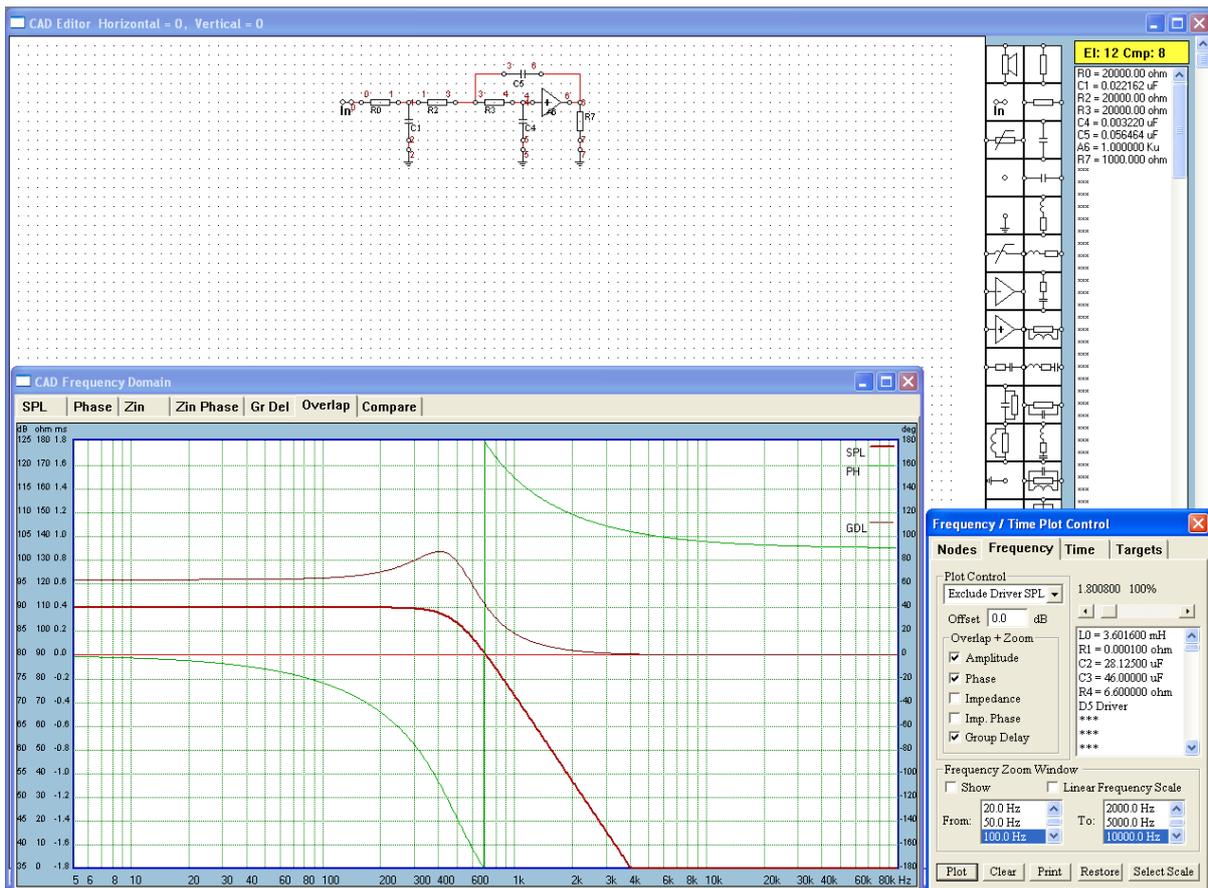
Fig 9.9 Butterworth -18dB/oct LP filter
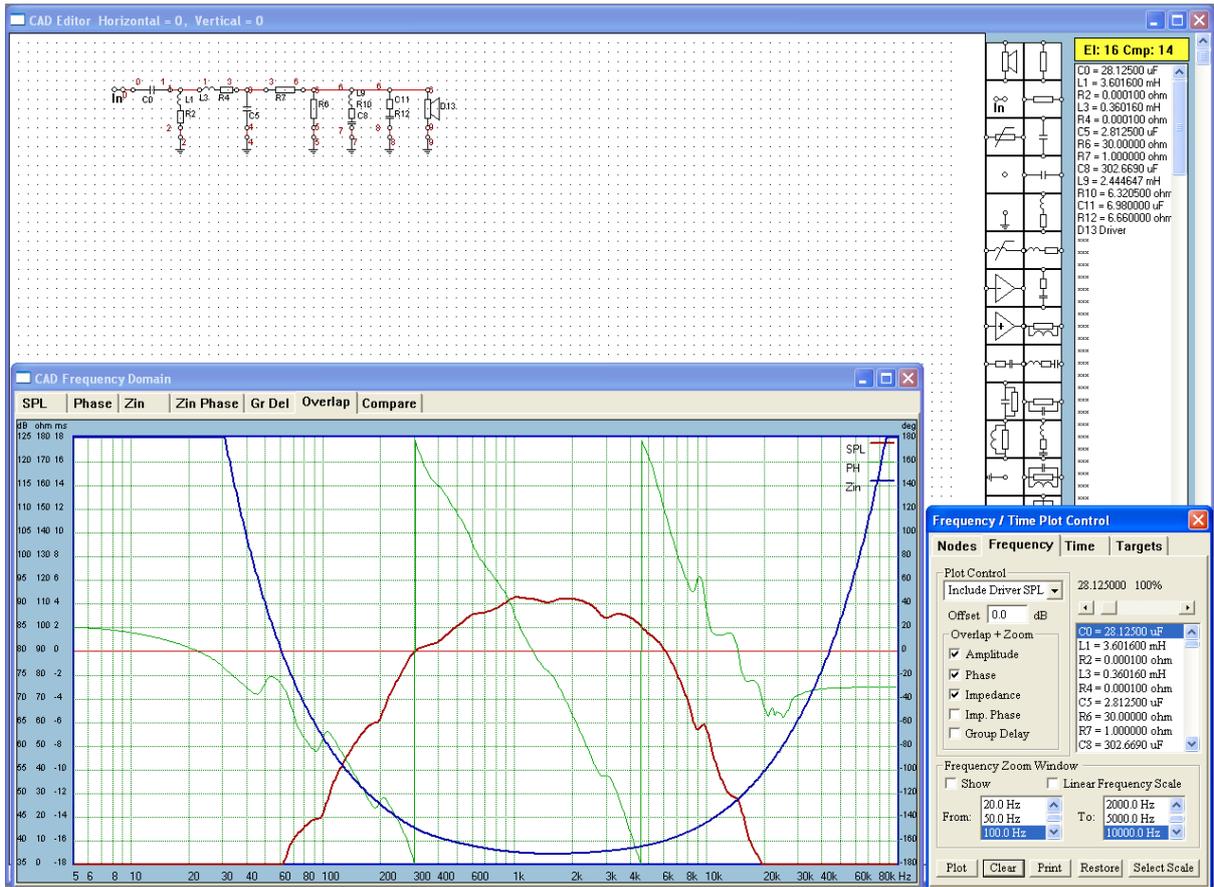
Fig 9.10 Example midrange driver.
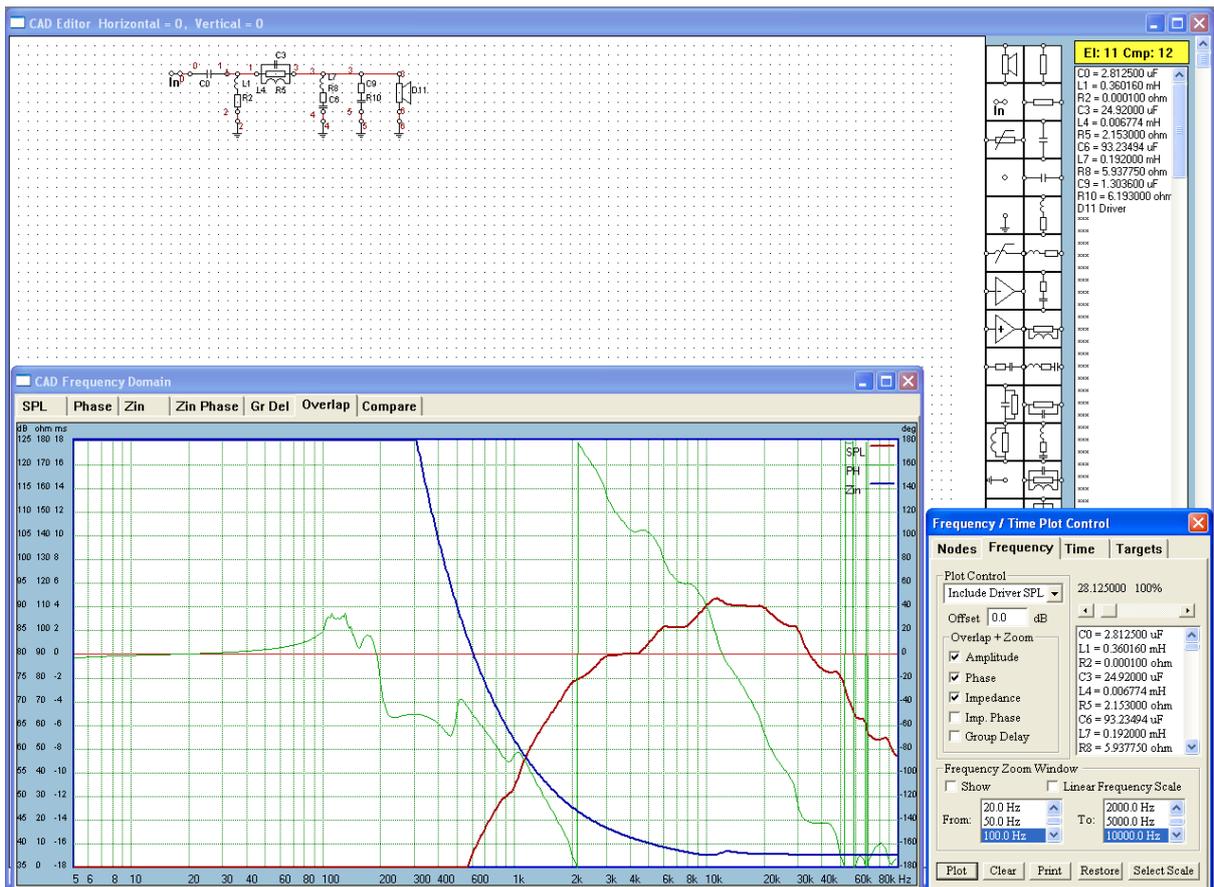
Fig 9.11 Example tweeter driver

**First Cut Design (Multiway & D'Appolito)**
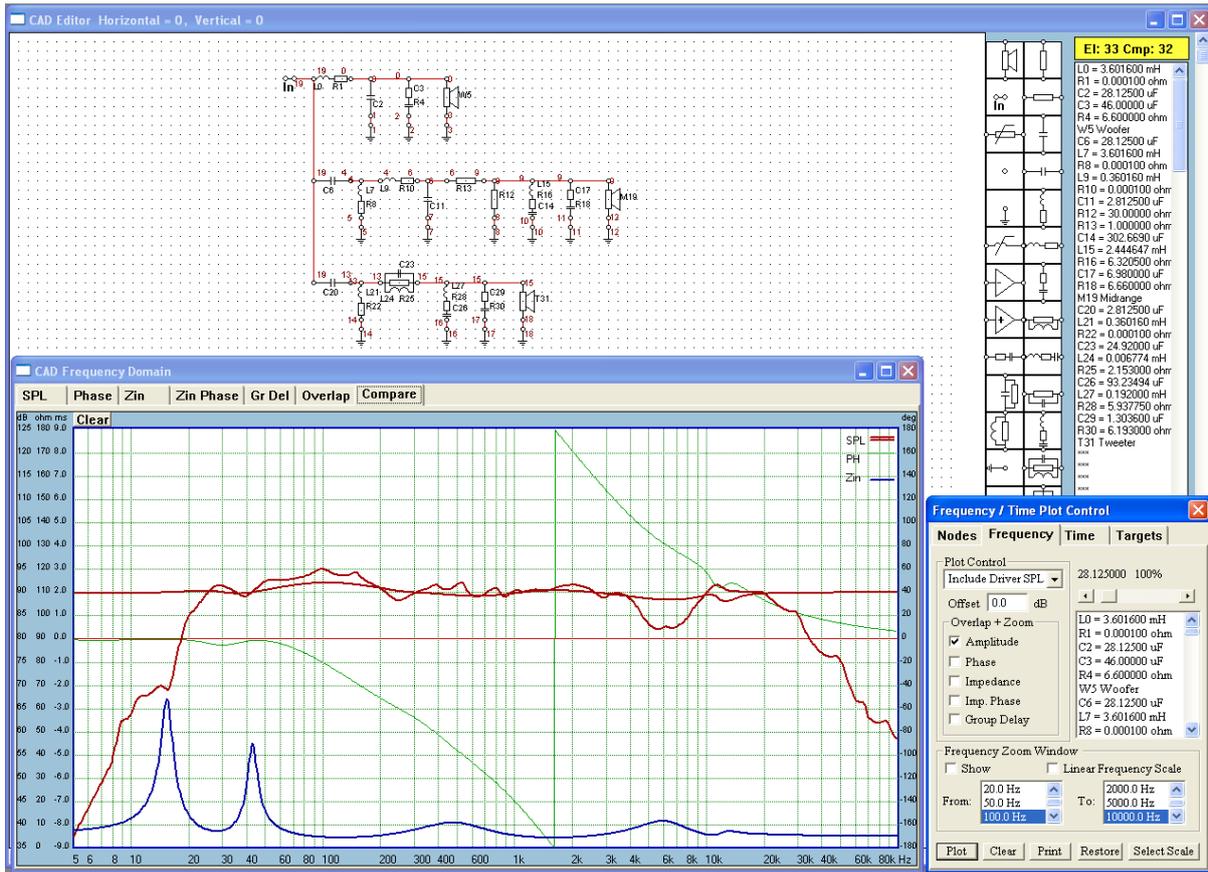
Fig 9.12 Complete 3-way crossover
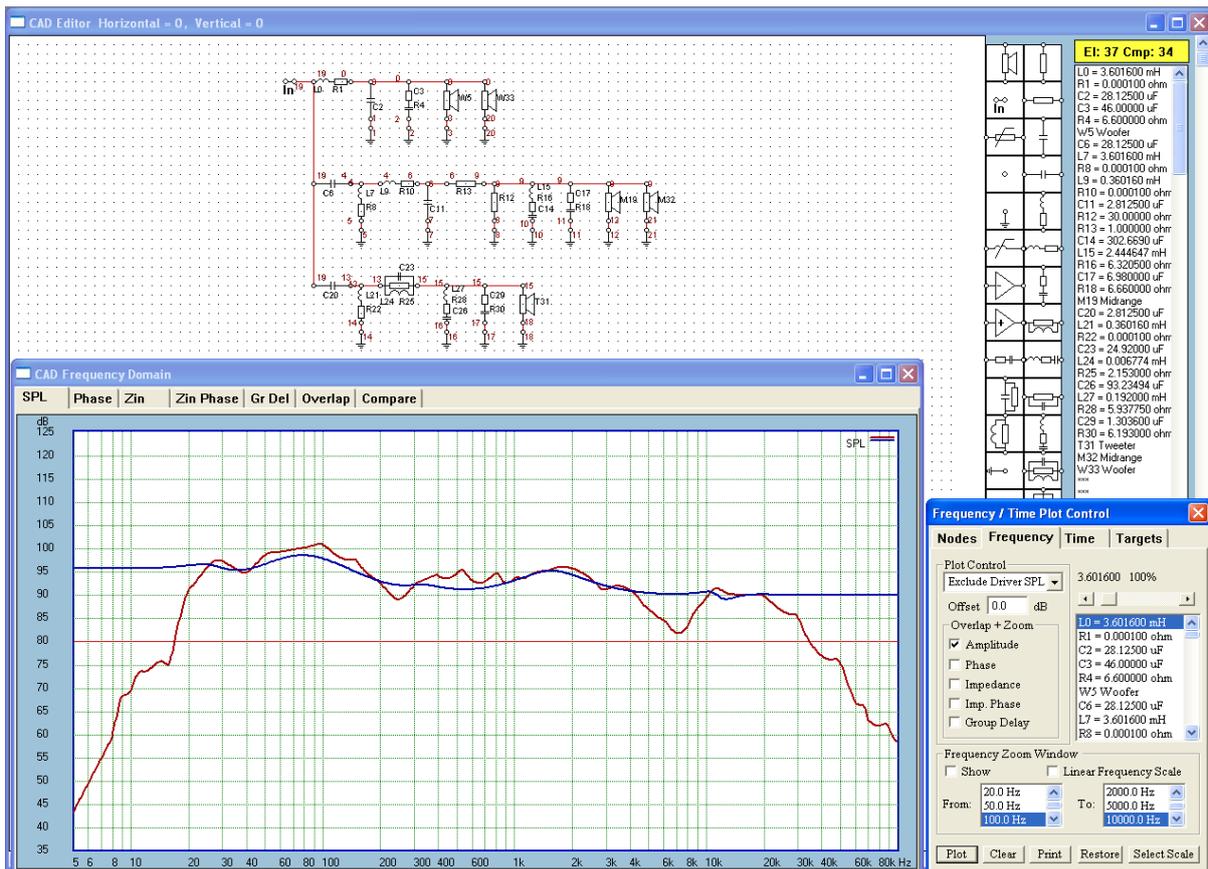
Fig 9.13 Complete WMTMW crossover

At the beginning of the design process it is often acceptable to create a "first cut" design, to check if basic ideas are feasible. To facilitate this concept, we have incorporated in the Crossover module a functionality, that allows you to create medium complexity, 5-way crossover and compensation network on one screen. The process is really no different from a single channel filter and you would proceed with the schematic as described before. An example of 3-way crossover is shown on Figure 9.12. You can plot frequency response of each individual channel or a summed response of up to 5 channels. This process is controlled from the dialogue box invoked just prior plotting. Additionally, you can load up to 5 driver files and include their frequency responses and input impedances for evaluating the complete system.

Here you must be aware of the fact, that no spatial (XYZ positioning) information is available at this moment to the program and all drivers are treated as co-located, radiating points. Full XYZ positioning is incorporated in the "System" module and you still need to use it for accurate modeling of the combined frequency response.

Also, you will able to optimize single channel or the complete crossover created this way.
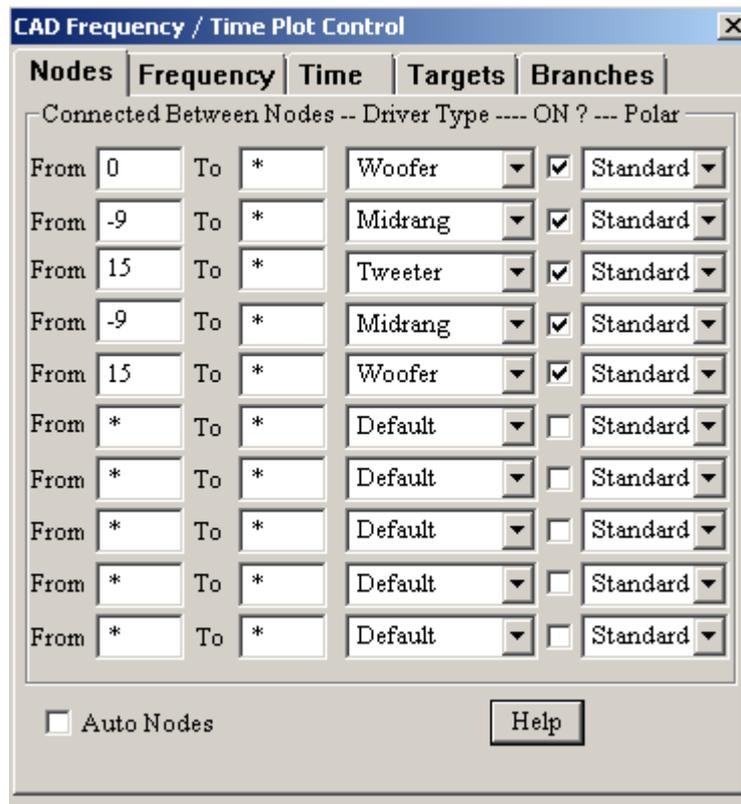
Figure 9.14. Example of node assignment for WMTMW configuration.

Working With Project Files

Project files are manipulated using "New Project", "Open Project" and "Save Project" options. These three options are designed to accelerate the loading of the three driver's files, keep them together as one project and facilitate tracking of your system design. This file is designed to hold the following:
1. Five paths and file names for the selected drivers.
2. XYZ mounting/tilting/rotating coordinates of each driver.
3. CAD schematic of the complete crossover.
4. System optimisation parameters and more…..

The file names are then used to automatically load the drivers' data files. The names of the drivers' data files can be entered into the project using the "New Project" option from the window menu. In summary, you would follow the steps below:

9.10

To create and save Project file:

1. Activate "New Project" menu under "File" main menu option. A dialogue box with 5 editable fields for path names will be opened. Please fill appropriate field for woofer, upper bass and press on "Load" button. The selected files will be loaded into the program's memory.
2. Save project file using "Save Project" option.

To load Project file:

1. Activate "Open Project" option.
2. Select and load required project.
3. You will be given an option to review the individual drivers' filenames.
4. Press "Load" button to load ALL the files into the program data space. The program will also re-open the default screen and will display all loaded drivers on the front baffle.

In summary:

1. Load all required driver files using "Open Project" option from the "Files" menu. This will ensure, that the frequency response and input impedance data is now available in program's memory and will be included in the plots. Please refer to further sections of this manual for information on using project files.
2. Create complete crossover and compensation network on one screen using CAD techniques described before. Your schematic should include loudspeaker symbols. We suggest, you save the schematic now – see Fig 9.12 or 9.13.
3. Assign driver type to each loudspeaker symbol. To accomplish this, double-click LEFT mouse button on each loudspeaker symbol. A selection box will be opened for you to choose on of the 5 available driver types. You can now select (enter a number 1 for woofer, ...5 for super tweeter) which driver is assigned to this symbol. Repeat the above for all drivers. This process assigns driver's input impedance to this particular schematic element (loudspeaker symbol) – see Fig 9.12 or 9.13.
4. Plotting control dialogue box (Fig 9.14) also contains additional fields for assigning drivers to nodes. Now, you need to **assign driver's frequency response to a particular node and this is accomplished by selecting driver's name from the associated list box.** An example of this process is shown Figure 9.12, where a 3-way crossover is developed.

1. Woofer (W31) is connected to Node 0 (N0),
2. Midrange (M19) is connected in reverse phase to Node -9 (N9) and
3. Tweeter is connected to Node 15 (N15).

Figure 9.14 shows correctly filled data fields in the plotting control dialogue box. You can observe, that system frequency response will be plotted using the following formula (midrange driver is connected "out of phase"):

$$\text{Frequency response = Node 0 - Node 9 + Node 15,}$$

Also, Node 1 has woofer driver assigned to it (woofer selected in the associated data field). Node 9 has midrange driver assigned to it (midrange selected entered in the associated data field) and Node 15 has tweeter driver assigned to it (tweeter selected entered in the associated data field). Figure 9.15 shows frequency response of each individual driver + filter and also total system frequency response. Switching ON/OFF of the individual channels is accomplished by inserting * (asterisk) instead of node number in the dialogue box shown on Fig 9.14.

If you just need to disable curve plotting of specific driver(s), you can also use "Do Not Plot" list box option from the same dialogue box.

Built-in Filters (Asymmetrical Band-Pass)

| Filter Type	+6/-XdB/oct	+12/-XdB/oct	+18/-XdB/oct	+24/-XdB/oct
Butterworth Passive	+6/-6dB/oct	+12/-6dB/oct	+18/-6dB/oct	+24/-6dB/oct
	+6/-12dB/oct	+12/-12dB/oct	+18/-12dB/oct	+24/-12dB/oct
	+6/-18dB/oct	+12/-18dB/oct	+18/-18dB/oct	+24/-18dB/oct
	+6/-24dB/oct	+12/-24dB/oct	+18/-24dB/oct	+24/-24dB/oct
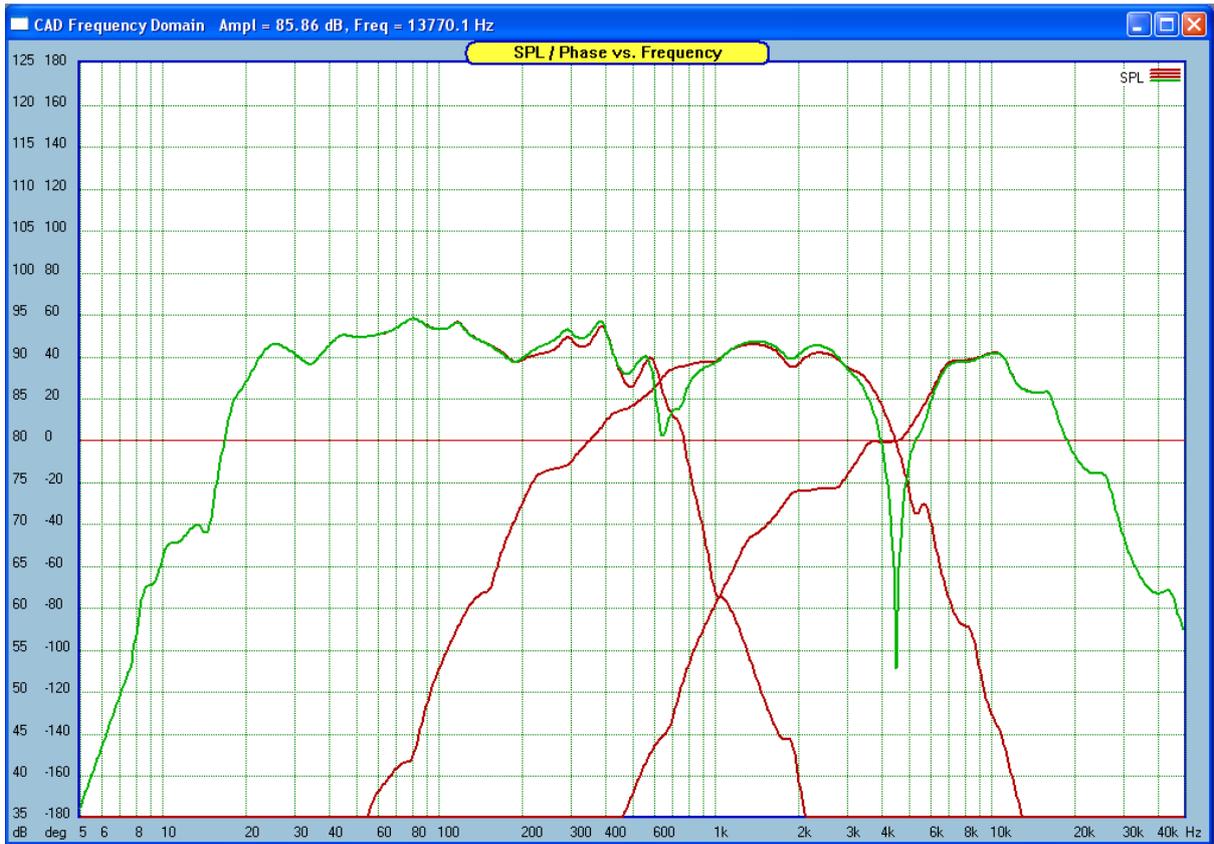
Figure 9.15 shows frequency response of each individual driver + filter and also total system frequency response of the system on Figure 9.12.

Filter Type	Low-Pass	High-Pass	Symmetrical Band-Pass
Bessel Passive	-12dB/oct	+12dB/oct	+/-12dB/oct
	-18dB/oct	+18dB/oct	+/-18dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct
	-30dB/oct	+30dB/oct	+/-30dB/oct
	-36dB/oct	+36dB/oct	+/-36dB/oct
Bessel Active	-12dB/oct	+12dB/oct	+/-12dB/oct
	-18dB/oct	+18dB/oct	+/-18dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct
Butterworth Passive	-12dB/oct	+12dB/oct	+/-12dB/oct
	-18dB/oct	+18dB/oct	+/-18dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct
	-30dB/oct	+30dB/oct	+/-30dB/oct
	-36dB/oct	+36dB/oct	+/-36dB/oct
Butterworth Active	-12dB/oct	+12dB/oct	+/-12dB/oct
	-18dB/oct	+18dB/oct	+/-18dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct
Bullock Passive	-6dB/oct	+6dB/oct	+/-6dB/oct
	-12dB/oct	+12dB/oct	+/-12dB/oct
	-18dB/oct	+18dB/oct	+/-18dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct
Linkwitz Passive	-12dB/oct	+12dB/oct	+/-12dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct
	-36dB/oct	+36dB/oct	+/-36dB/oct
Linkwitz Active	-12dB/oct	+12dB/oct	+/-12dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct

All-Pass Passive	-12dB/oct	+12dB/oct	+/-12dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct
			+/-6dB/oct
			+/-18dB/oct
Notched	-24dB/oct	+24dB/oct	

Built-in Crossovers

Filter Type	2-Way	3-Way	4-Way	5-Way
Bessel Passive	-6dB/oct	+6dB/oct	+/-6dB/oct	+/-6dB/oct
	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
	-18dB/oct	+18dB/oct	+/-18dB/oct	+/-18dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct	+/-24dB/oct
Bessel Active	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
Butterworth Passive	-6dB/oct	+6dB/oct	+/-6dB/oct	+/-6dB/oct
	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
	-18dB/oct	+18dB/oct	+/-18dB/oct	+/-18dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct	+/-24dB/oct
Butterworth Active	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
Linkwitz Passive	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
	-24dB/oct	+24dB/oct	+/-24dB/oct	+/-24dB/oct
Linkwitz Active	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
Butterworth type 1 Passive -12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct	
	-24dB/oct	+24dB/oct	+/-24dB/oct	+/-24dB/oct
Butterworth type 1 Active	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
Notched	-24dB/oct			
Bullock Pasive	-6dB/oct	+6dB/oct	+/-6dB/oct	+/-6dB/oct
	-12dB/oct	+12dB/oct	+/-12dB/oct	+/-12dB/oct
	-18dB/oct			
	-24dB/oct	+24dB/oct	+/-24dB/oct	+/-24dB/oct

Built-in Networks

Network Type

Passive Lattice Time Delay Networks
Linkwitz-Riley Transformer
Time Delay – First Order Circuit
Transient Perfect (TP) second order
Butterworth Q/Fo Type
Active Q-Boost
3 types of Bi-Quad circuits (HP, LP, BP)
Lintwitz & Butterworth asymmetrical 12/24dB filters.

Compensation of Diffraction Distortion

As we discussed before, diffraction distortion is best observed in the anechoic chamber. Here, the loudspeaker radiates into full surrounding space (4π radiation). However, portion of the sound is reflected from the cabinet edges and combines with the direct sound at the observer location. Resulting SPL depends on the frequency and total geometry of design. An example of this process is shown on Figure 9.16, where the front panel dimensions are X=40cm and Y=60cm and the driver is located towards the bottom of the panel. It is observable, that SPL starts rising quite early, at 50-60Hz region for this design, reaches +6dB "gain" at 300Hz and continues at this level developing small bumps.

Compensating for this frequency response distortion revolves basically around three options: (1) acoustical attenuation of the sound travelling towards the edges of the cabinet. The aim here is to prevent the diffraction phenomenon at the first place.
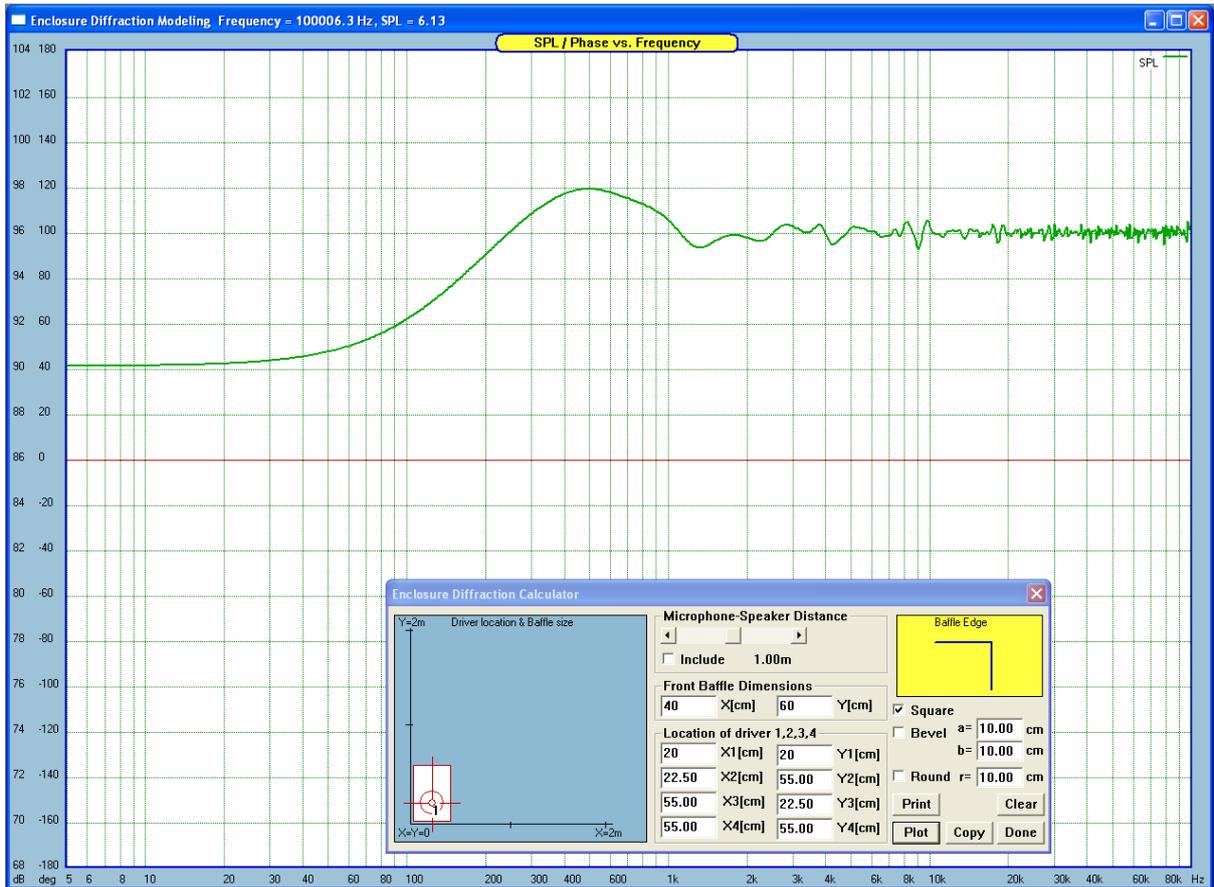
Fig 9.16 Diffraction distortion

9.14
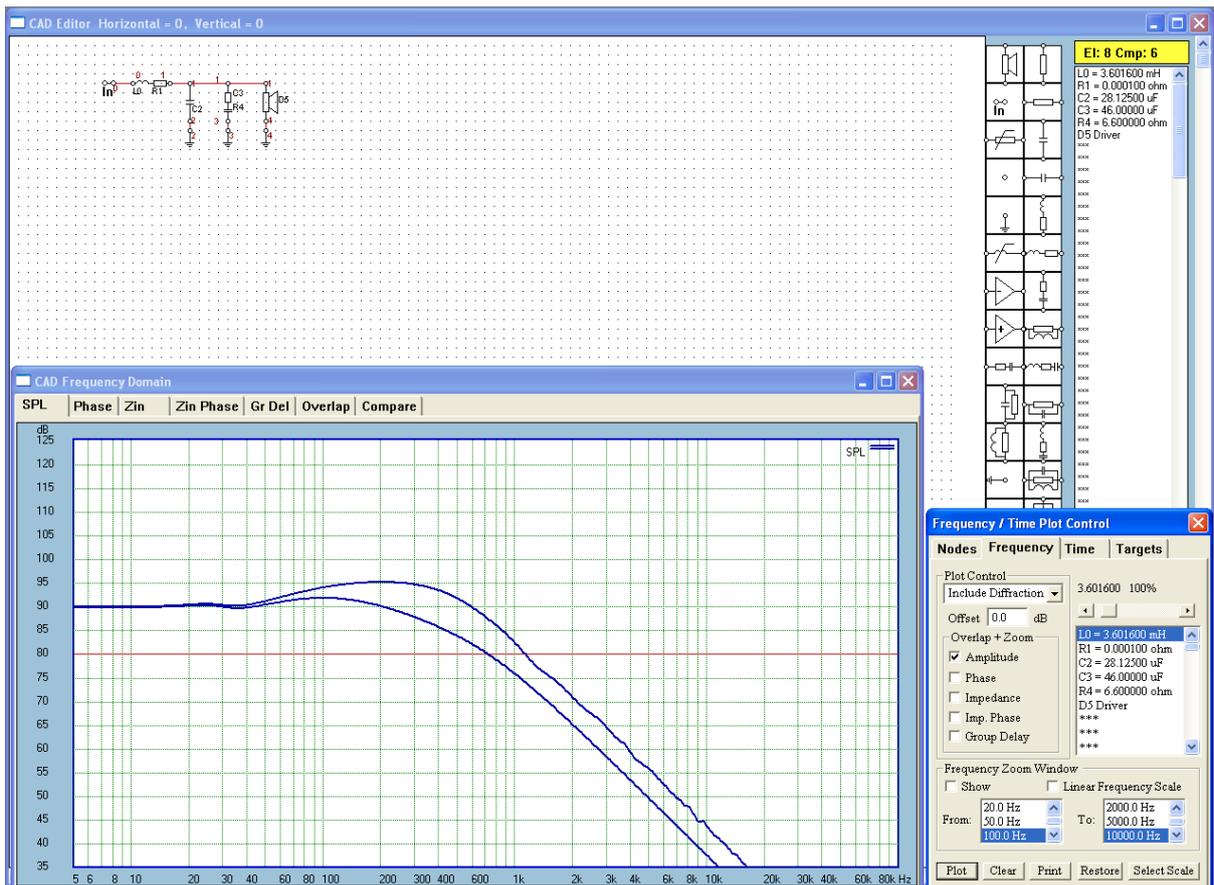
Good results have been reported by placing sound absorbing material on the front panel around drivers and making the cabinet edges rounded. (2) D'Appolito configuration. By adding second woofer on the front panel, the low frequency SPL gain can be as high as +6dB. Once again, by careful selection of all contributing system parameters, one can reduce the +6dB diffraction step. Finally, (3) electronic compensation. Figure 9.17 shows two curves (1) crossover frequency response when connected to a real driver as a load impedance - Network Amplitude and (2) as before + diffraction loss. The extra SPL gain due to the diffraction needs to be compensated.

Electronic compensation could be arranged as passive or active circuit. However, if the passive option is chosen, you need to be aware of possible problems associated with this option. Our problem could be stated as finding a way to reduce system gain within certain frequency range. Therefore, our compensation circuit needs to contain reactances (capacitors and inductors). These additional reactances will combine with the crossover's reactances and most importantly, with already existing loudspeaker input impedance. Therefore, the resulting frequency response will be a compromise between reducing the diffraction distortion and introducing additional ripple into the frequency response. This problem is illustrated on Figure 9.18, where the +6dB diffraction step has been equalized, at the expense of +2/-1.5dB ripple below 100Hz. The circuit used was a parallel tank circuit (R7‖L6‖C8) and also low-pass crossover component (L0) value was modified.

Somewhat better result was obtained when active compensation circuit was implemented. Figure 9.19 shows two curves: the frequency response of the compensation circuit itself (measured at Node 3 - Excluding Driver) and the total system frequency response (measured at Node 5 - Include Enclosure Diffraction).

In order to include the enclosure diffraction effect in the graphs, please select "Include Encl. Diffr." option from the list box of the plotting screen. The plots will not contain driver's acoustic frequency response.

**Also, the diffraction distortion gain is NOT included in the crossover optimization process.**
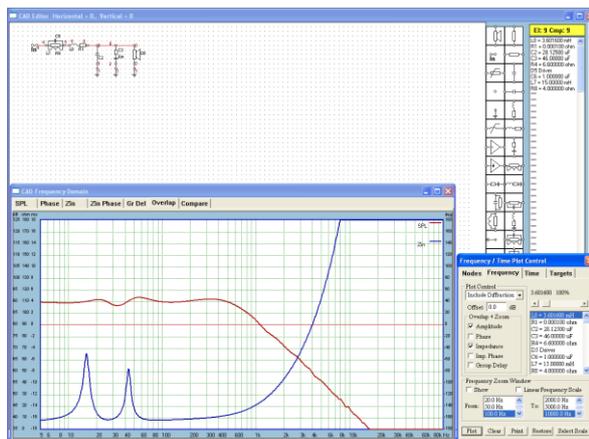

Fig 9.18 Passive equalization of diffraction   Fig 9.19 Active equalization of diffraction.
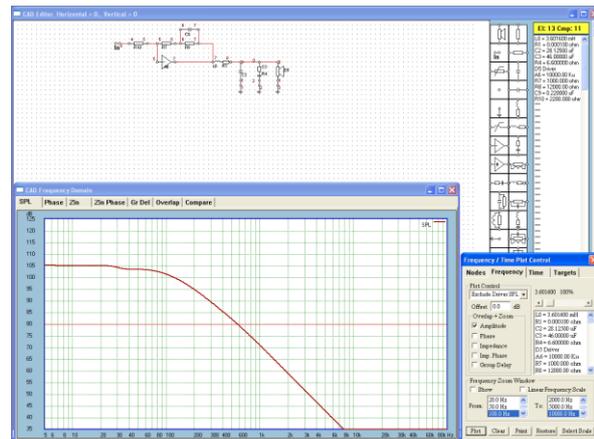
Passive Time Delay Circuits

Current SoundEasy release also incorporates two passive time delay circuits: First Order Lattice and Second Order Lattice. Both networks are accessible from the "Built-in" filters dialogue box.
1. Open the **Edit -> "Built-in Filter"** dialogue box.
2. Select **"Passive Lattice 1-st" or "Passive Lattice 2-nd"** from the "Filter Type" list box.
3. This will invoke **"Lattice Network Calculator"** dialogue box.
4. Enter the required **"Design Parameters"** and click on **"Exit".**
5. Once you press the Left Mouse Button above the CAD screen, the Lattice Network will loaded and displayed on the screen.

You can easily cascade the networks to obtain larger delay. An example of three cascaded time delay, second order networks are shown on Figure 9.20. Please remember to keep only one Input Node in the circuit. As you can see, with the exception of terminating resistor, R36, load resistors in the preceding networks were replaced by the consecutive lattice network. Plot "0" – Nodes: 1 and -2, Plot "1" – Nodes: 4 and –5, Plot "2" – Nodes: 6 and –7.
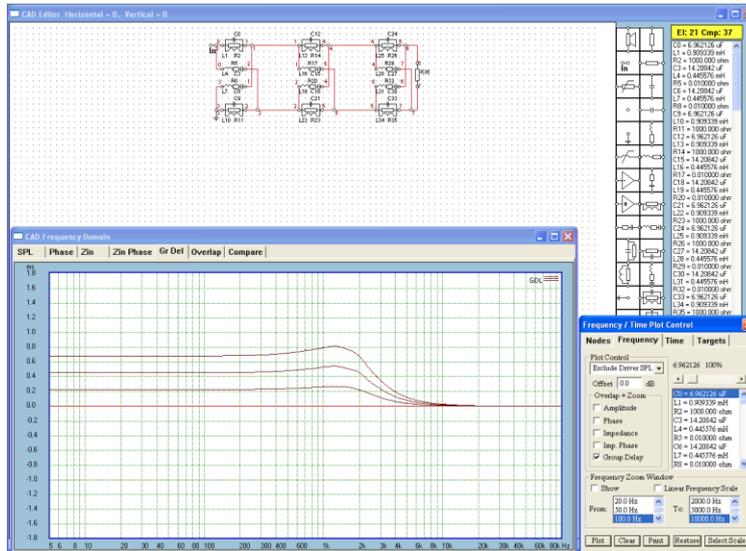
Figure 9.20. Example of cascaded Second Order Lattice and their Group Delay plots.

Multi-way Transient Perfect (TP) Crossovers

Current release of the program provides you with a full support for design and optimization of passive and active, multi-way Transient-Perfect (TP) crossovers. The TP crossovers are a class of pre-distorted filters with suitable modified frequency response, so that a square wave run through such a design, will maintain its perfect shape after the summation. Here is a short example:

Woofer: Low-Pass,   Fc=500Hz,   Node 7
Midrange: Band-Pass,   500-5000Hz,   Node 18
Tweeter: High-Pass,   Fc=5000Hz,   Node 21
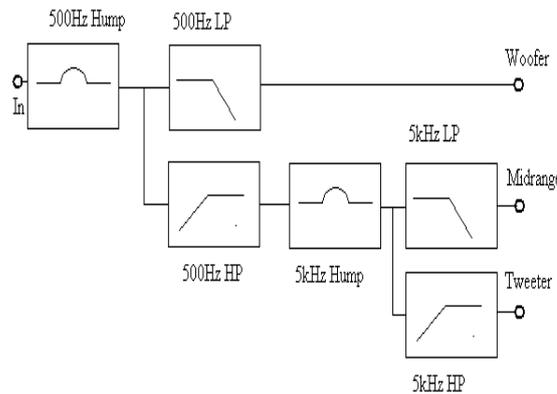
Figure 9.21 3-Way TP Crossover block diagram
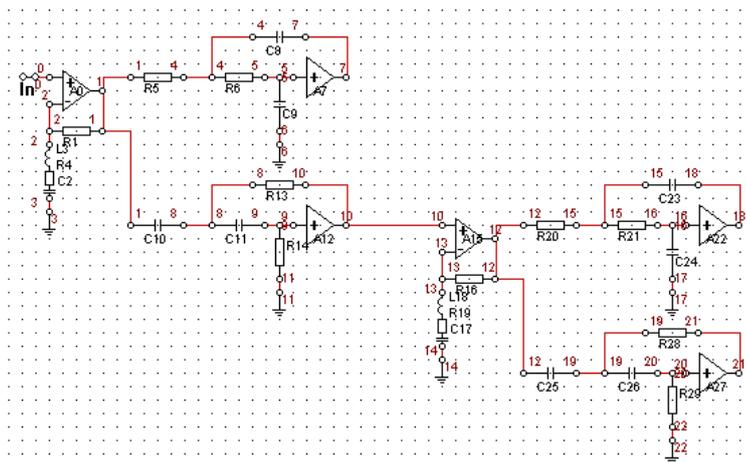
Figure 9.22. 3-Way TP Crossover

9.16

Principle of creating multi-way TP networks:

Cascade another 2-way TP network (Hump+LP+HP) from tweeter port. Therefore, 4-way, 5-way… crossovers can be assembled the same way.
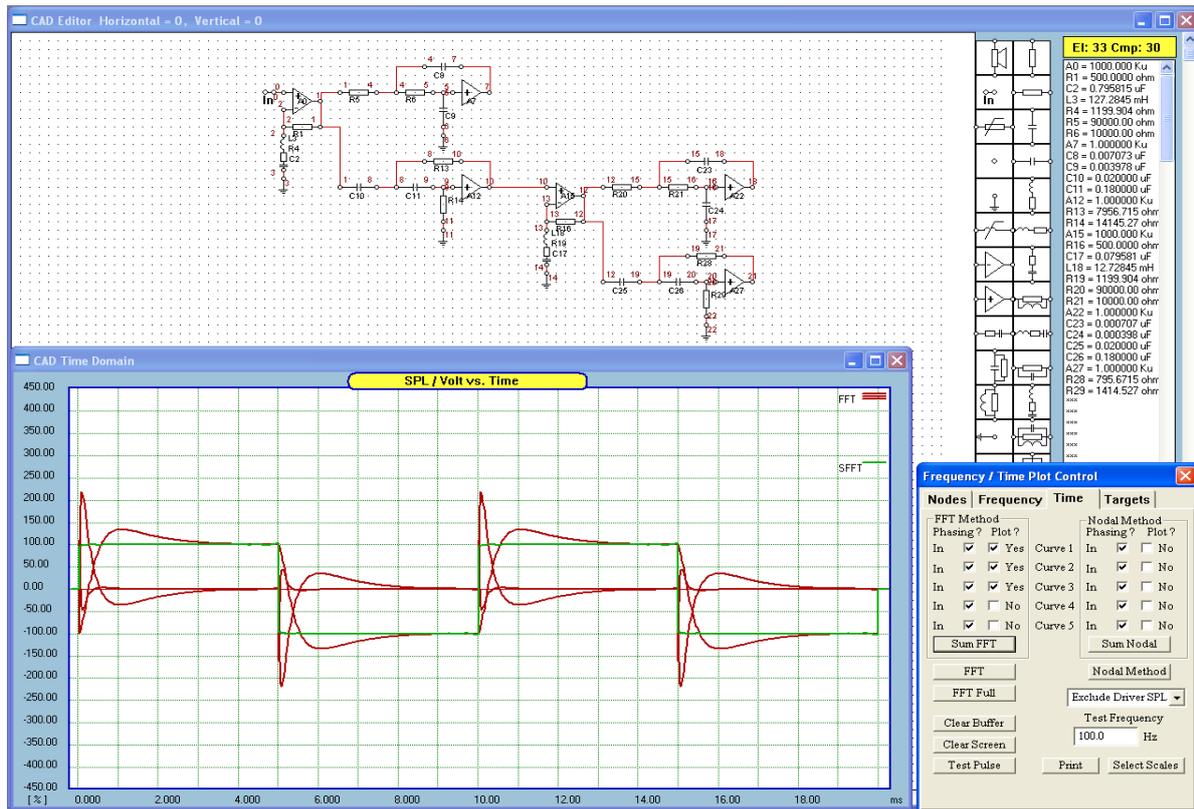
Figure 9.23 Green color is the summed time response of 3 channels – perfect square wave
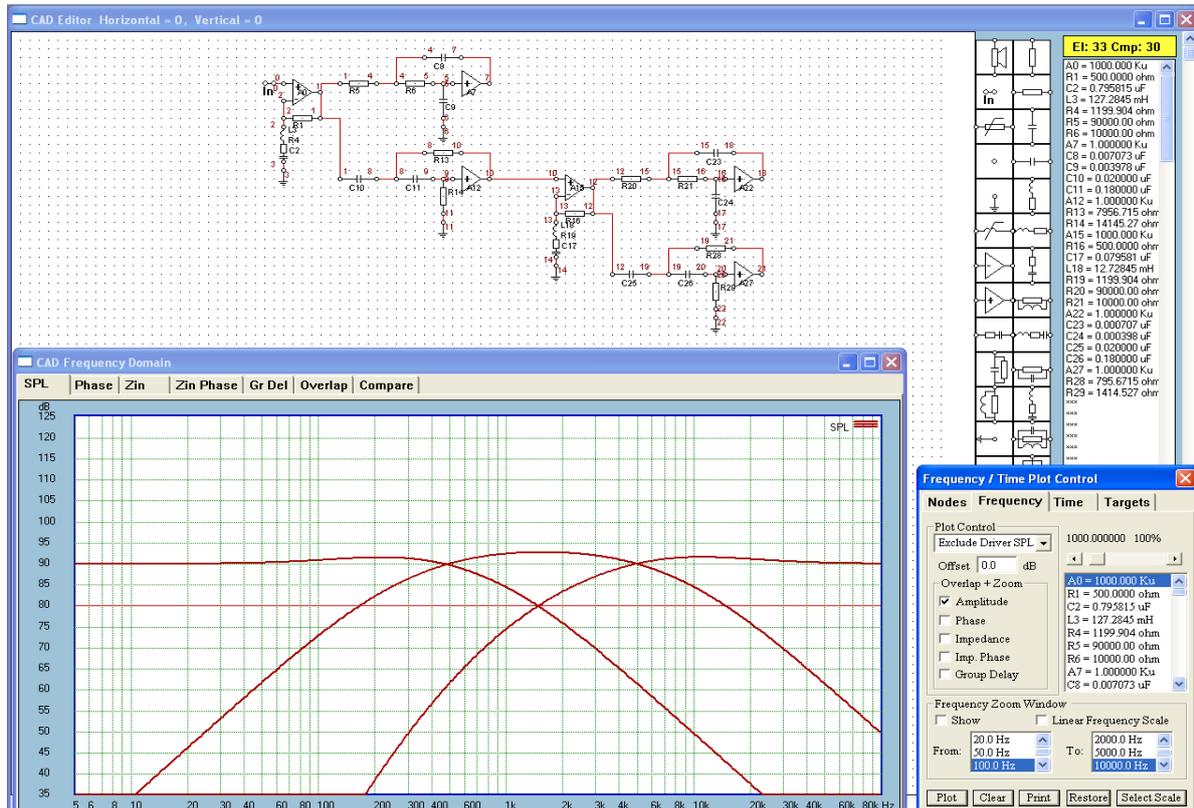
Figure 9.24. Red – SPL – individual channels.

Additional requirements:
1. The active implementation of the 3-way TP network requires three (3) power amplifiers to be connected to each of the crossover outputs. Obviously, 4-way TP network requires 4 amplifiers and so on….
2. Gain of the each channel (power amplifier + speaker's SPL) MUST be set exactly the same.
3. 2-way TP HP/LP filters can be implemented as passive networks, therefore you can get away with single power amplifier.

SoundEasy 2-nd order TP Calculator is shown below. The 2-way, 2-nd order active crossover with EQ correction is your building block. All you need to enter is crossover frequency and overlap parameters. Filter Section parameters and Equalizer Section parameters are calculated automatically from the two mentioned above. However, you can still edit Filter and Equalizer parameters to force the program into "what-if" analysis. Targets for HP and LP sections of the TP crossover are also built-in for optimizations of the full acoustic response of the crossover – see figure below.

Acknowledgement:
The 2-way TP Crossover and Calculator concepts are due to the excellent papers from John Kreskovsky.
(www.geocities.com/kreskovsky/John1/html)
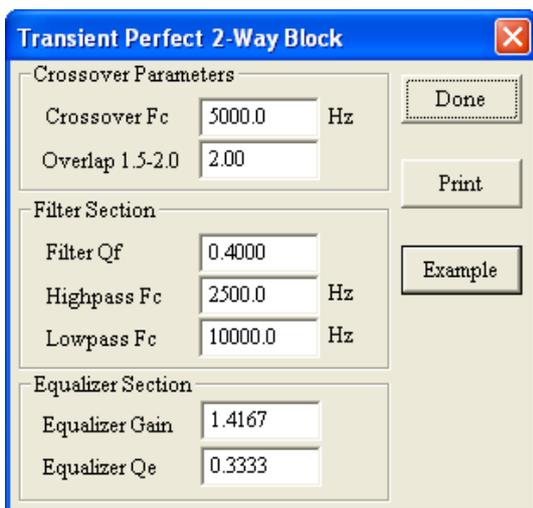
Figure 9.25
2-nd order TP Calculator – your "building block"
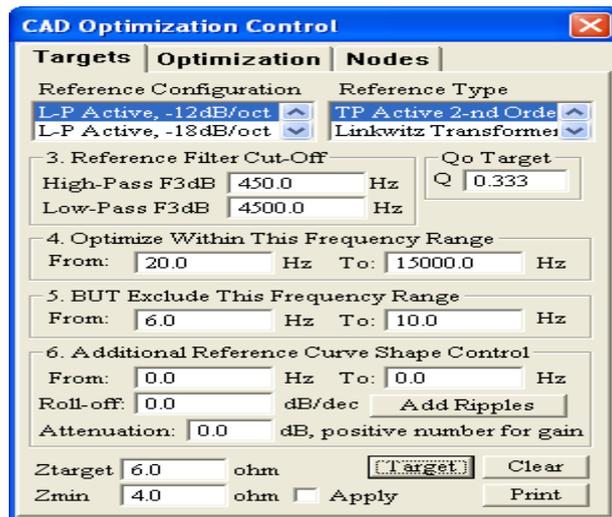
Figure 9.26
HP and LP Optimizer templates for TP, 2-nd order.
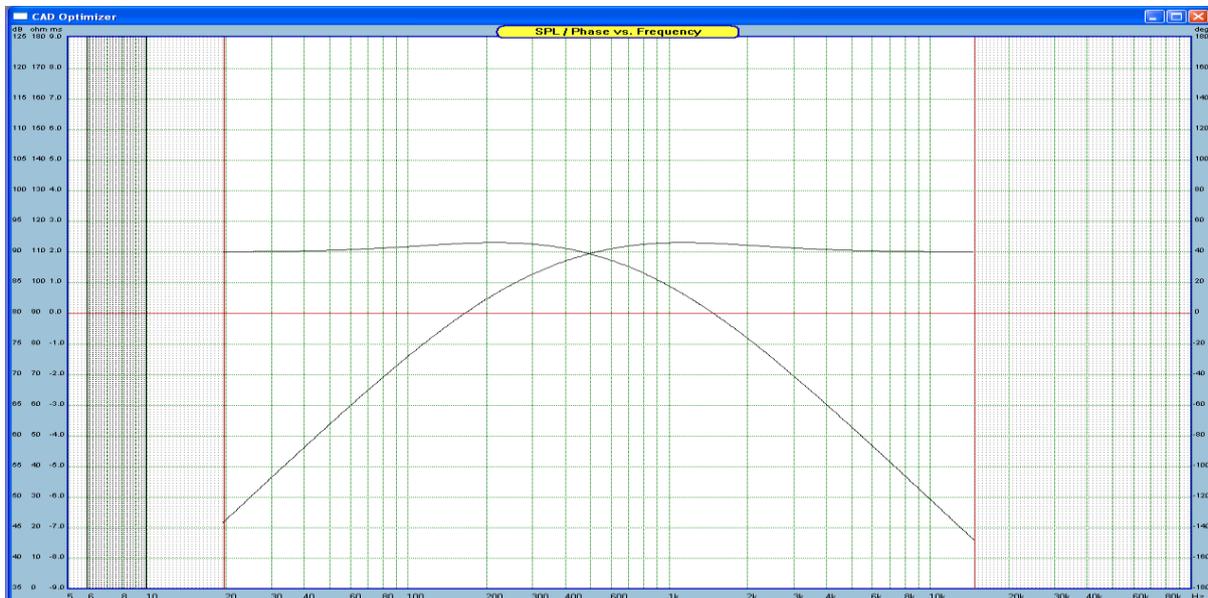
Figure 9.27. Optimizer TP reference curves example.

**TP 2-nd order crossover version with GYRATOR replacing inductor in the EQ circuit.**
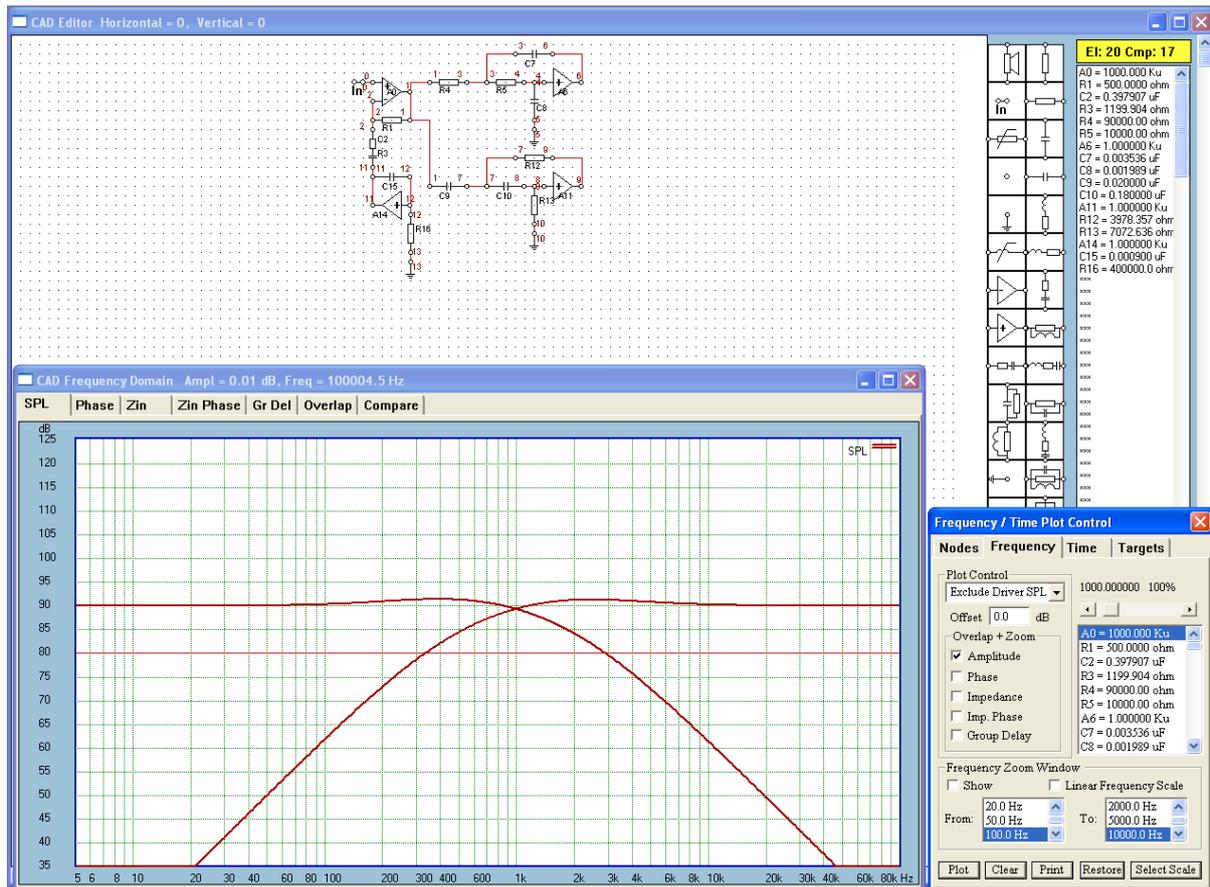
Figure 9.28. TP 2-nd order crossover version with GYRATOR replacing inductor in the EQ circuit

Note: A14 output impedance is the used as one of the GYRATOR's components. Outputs are Node 5 (LP) and Node 9 (HP).

Approximated value of the inductor, L, created with the gyrator:

L = (R17 – Rout)*Rout*C15
Rout = 200 ohm, output impedance of the A16. Rout should be selected from 200-470 ohm.
R17 = 400 kohm
C15 = 0.009 uF
Hence: L = 70mH

Approximated value of the inductor's, Q, created with the gyrator:

Q = XL/(Rout+R4)
Rout = 200 ohm
R4 = 1100 ohm
L = 0.070 H
Hence: Q = 0.33

Approximated value of the gain, G, created with the gyrator:

G = 1 + R2/(Rout+R4)
Rout = 200 ohm
R4 = 1100 ohm
R2 = 500 ohm
Hence: G = 0.385

Digital Filter

There are several methods of implementing a digital filter on your PC. One that looks attractive and is simple to use would possibly involve:

1. Connecting your CD-player to the sound card input,
2. Connecting several identical power amplifiers + loudspeaker drivers to the sound card outputs,
3. Generating required filter transfer functions – using CAD screen and
4. Playing music through the sound card – now your "digital crossover".

Set up described above, requires your to purchase and measure drivers and actually built the enclosures, therefore the procedure significantly increases costs of your design process and really goes well beyond computer modelling.

There are two basic areas where experimenting with digital filters may be of advantage:

1. Quick comparison of filter topologies. You may be interested in actually listening to a sub-woofer channel with different crossover orders or type or EQ-circuit.
2. Equalizing driver channel response. For instance, your goal is to produce **acoustical response of the driver channel that is identical to 4-th order LR filter**. What is needed in this case, is the digital equalizer, that will correct the imperfect driver's response (ALL small SPL irregularities) to the ideal 4-th order LR filter.

In any case, you must exercise extreme care with amplifiers in order to avoid potential damage to drivers from incidental signals that are outside their intended frequency range.

To accomplish the above, the program implements important DSP technique, **the overlap-add method, and FFT convolution**. The overlap-add method is used to break long time domain signals into smaller segments for easier processing. FFT convolution uses the overlap-add method together with the Fast Fourier Transform, allowing signals to be convolved by multiplying their frequency spectra. For longer time domain samples, the FFT convolution tends to be faster than standard convolution, while producing exactly the same result.

FFT convolution uses the principle that multiplication in the frequency domain is equivalent to convolution in the time domain. The sampled 8192-sample blocks of input signal is transformed into the frequency domain using the FFT, multiplied by the frequency response of each filter, and then transformed back into the time domain using the Inverse FFT. By using the FFT algorithm to calculate the Discrete Fourier Transform, convolution via the frequency domain can be **faster** than directly convolving the time domain signals. The final result is the same; only the number of calculations has been changed by a more efficient algorithm. For this reason, FFT convolution is also called **fast convolution**.

The overlap-add method is based on the fundamental technique in DSP: (1) decompose the input signal into smaller blocks, (2) process each of the blocks using fast convolution, and (3) recombine the processed blocks into the output, time-domain signal. Fourier Transform treats our 8192-sample block of input data, X(t) and the filter impulse response H(t) as though they are periodic. Convolution treats them as thought they are not. In order for convolution theorem to be of any use in our case, we need to either perform the convolution in a "circular" manner (which often causes problems at the edge of the processed blocks), or pad the blocks with zeros, so that "circular convolution" becomes "linear convolution".

We anticipate to use the FFT routine to build our "block filter" because it requires far fewer calculations than a conventional FIR filter for long impulse responses. For example: For an impulse response of length N, an FIR filter does N^2 calculations for each block of N input samples. A block FFT filter has to do 2 FFTs and N multiplications for each block of N input samples. This is $2(N*\log_2 N)+N$ calculations. If N=1024, then the FIR filter does $1024^2 = $ **1 million calculations** for each input block, while the FFT filter does $2(1024*10)+1024 = $ **21 thousand**. However, the problem with our "Block filter" right now is the circular convolution operation does not exactly match the linear convolution operation. In practical application, you would hear quite loud "clicks" from the loudspeaker when using circular convolution-based block filter.

The method described here is a popular way to overcome the problems with circular convolution. There are actually two popular methods: the overlap-and-add method and the overlap-and-save method. Presented below is only the first one. In our case, the basic difference between linear and circular convolution is that the block of the 8192-sample input signal is appended with a 8192-long string of zeros. The total length of the input block of data is now 16384-samples. These zeros allowed the last impulse in the input signal to generate it's impulse response without wrapping around to the start of the signal. The block FFT filter we are trying to design, should do the same thing.

Another thing worth noticing, is that we can save some computational time for each block of input data, because the filter transfer function, H(f) does not change for each block. We can calculate it once, store and re-use for all input blocks of data. Please also notice, that the program does not need to start calculating the filter transfer function as the FFT of its impulse response padded with N-zeros. We already have the filter's transfer function represented in frequency domain. Listed below is a step-by-step algorithm for an impulse response that is N samples long:

1. Calculate H(f) – this is actually done by the CAD screen functions – zero padding "equivalent" is performed in frequency domain. The H(f) is now 2N long.
2. Read-in N samples from the input waveform device of the soundcard.
3. Pad the N samples with N zeros – now, the sample length is 2N long.
4. FFT the input block of samples, and designate it as X(f) – FFT is now 2N long.
5. Multiply X(f) and H(f) to obtain Y(f) – there are 2N multiplications.
6. IFFT Y(f) to get the time domain signal, y(t) - the IFFT is also 2N long.
7. Divide y(t) into 2 blocks, the N1 and N2 samples, N1=N2.
8. The N1 samples are added to the N2 samples from the previous block, then written out to the soundcard.
9. The N2 samples of current y(t) are stored for re-use in the next block.
10. Go back to step 2 for next block of input data.

For "linear convolution" the formula: $2(N*\log_2 N)+N$ needs to be modified. Firstly, after zero-padding, the FFTs are performed over 2N blocks of data. Secondly, the number of multiplications is also doubled to 2N. Therefore, the final formula is:

$$2(2N*\log_2(2N)) + 2N = 2(2N*(\log_2(2) + \log_2(N)) + 2N = 4N(1 + \log_2(N)) + 2N = \mathbf{6N + 4N* \log_2(N)}$$

Taking into account the figure (N=1024) from the previous example: 6*1024 + 4*1024*10 = **46 thousand**. This is still much less than **1 million calculations** for each input block processed with FIR filter. The method described above was implemented in the program.

Operation of the Digital Filter Dialogue Box

Before you can use the Digital Filter function, you need to have the filter or crossover completely designed on the CAD screen. This step is no different from your normal CAD activity. You should also plot the amplitude response of the filter to make sure, that design works correctly. Also, if your filter has too much gain, you will drive the sound card output DAC into saturation. **We strongly recommend, you should start slowly and familiarize yourself with the methodology and the set-up quite carefully**. A good starting point is a simple RC low-pass filter with –6dB/oct slope.

Now, its time to connect the amplifiers to the sound card outputs and your CD-player to the input of the sound card. Please note, that there is no volume control on the input of the sound card, therefore, you need to make sure, that the sound card will not be driven into clipping. The digital filter uses "Line-in" input of the sound card, so chances are, that CD-player output voltage matches the input sensitivity of the sound card "Line-in" input. The volume control on the "Digital Filter" dialogue box is only used to control output level from the sound card. Again, please be careful not to overdrive the output DAC of the sound card, as sever clipping will occur. To make things worse, your loudspeakers are now connected **directly** to amplifiers outputs', so any clipping distortion may cost you dearly.

Before you can play music through the sound card, you need to have the crossover transfer function calculated. To accomplish this, press the **"2. Calculate Filters + Link Outputs" button**.

To start the filtering process, simply press **"3. Start Filter"** button and to end the process, press the **"4. Stop Filter"** button. If your sound card is suitable for this function, there will be no warning messages and the sound will appear at the output of the sound card.   Here is the functionality of all controls:

1. **Drivers Group**
    - **Dr1** – Check box to turn ON driver connected to the first pair of nodes in the schematic.
    - **Dr2** – Check box to turn ON driver connected to the second pair of nodes in the schematic.
    - **Dr3** – Check box to turn ON driver connected to the third pair of nodes in the schematic.
    - **Dr4** – Check box to turn ON driver connected to the fourth pair of nodes in the schematic.
    - **Dr5** – Check box to turn ON driver connected to the fifth pair of nodes in the schematic.
    - **Delay [ms]** – Enter required delay for each driver.

2. **Sampling Rate** – Select 48kHz OR 44.1kHz ONLY.
3. **Outputs** – You can redirect crossover filter outputs to a designated sound card output. For full information, please see "Redirecting Outputs" in the following sections.
4. **Calculate Filters + Link Outputs** – Press this button to calculate and plot SPL /Phase of your individual filters (channels) in the crossover. You must have your filter transfer function calculated first, before you can start filtering the music. Every time you change the assignment of crossover outputs to the outputs of the sound card, you MUST activate the new links by pressing this button.
5. **Start Filter** - Press this button to start Digital Filter. You should now hear the sound in each channel, filtered by your crossover.
6. **Stop Filter** – Press this button to suspend the Digital Filter function.
7. **Plot System** – Press this button to plot total crossover response.
8. **Clear** – Press this button to clear plots.
9. **Done** – Press this button to close the dialogue box and exit.
10. **Select Curve For Plotting Group**
    - Phase – Check this box for plotting filter and system phase.
    - Amplitude – Check this box for plotting filter and system SPL.
11. **Volume** – Use this slider to adjust sound card output.
12. **Mono Group**
    - **1x2 way to 1x4 way** – Check one of these boxes to set the sound card to required mono resolution.
13. **Stereo Group**
    - **2x2 way to 2x4 way** - Check one of these boxes to set the sound card to required stereo resolution.
14. **Component's List Box** – Double-click on the component you need to adjust.
15. **Component Adjust Slider** – Use this slider to change component's value from 1% - 1000%.
16. **Restore Component** – You can restore the original component's value by pressing this button.
17. **Project 1** – Press this button to make Project 1 current and active.
18. **Project 2** – Press this button to make Project 2 current and active.
19. **Project 3** – Press this button to make Project 3 current and active.

You can add **Delay to the signal path** for each driver be entering delay in milliseconds in the provided data entry fields. Corresponding physical offsets will be displayed in the "Offset" column. You can also change the component value using the slider, while the DF is active. Switches Dr1 – Dr5 will turn ON/OFF drivers and the processed signal.   When you start the Digital Filter, it is most preferred, that you limit yourself to changing component values only. This is to minimize sound breaks introduced by forcing your computer to deal with additional CPU tasks.

You may also notice, that Digital Filter plotting window can not be customized – for the same reason – to minimize CPU usage. If you need to change the size or location of the plotting window, please Stop the filter first, close the control box and select "Digital Filter Plotting" main menu option. Now, you can change the parameters of the window, and they will be "remembered" when you use Digital Filter next time.
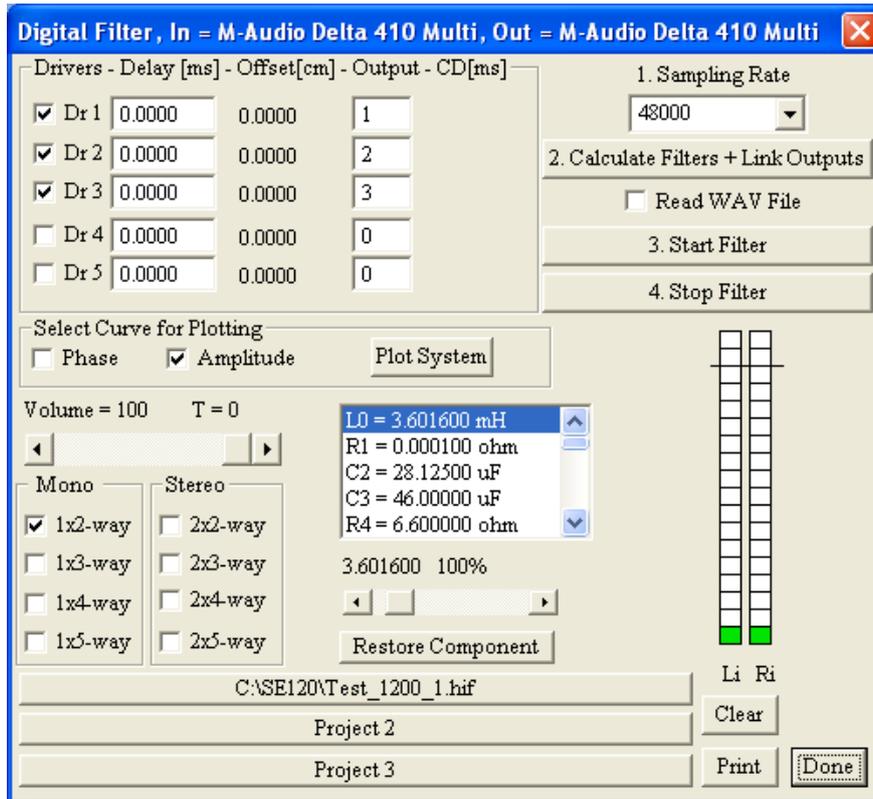
Figure 9.29. Digital Filter Control Box
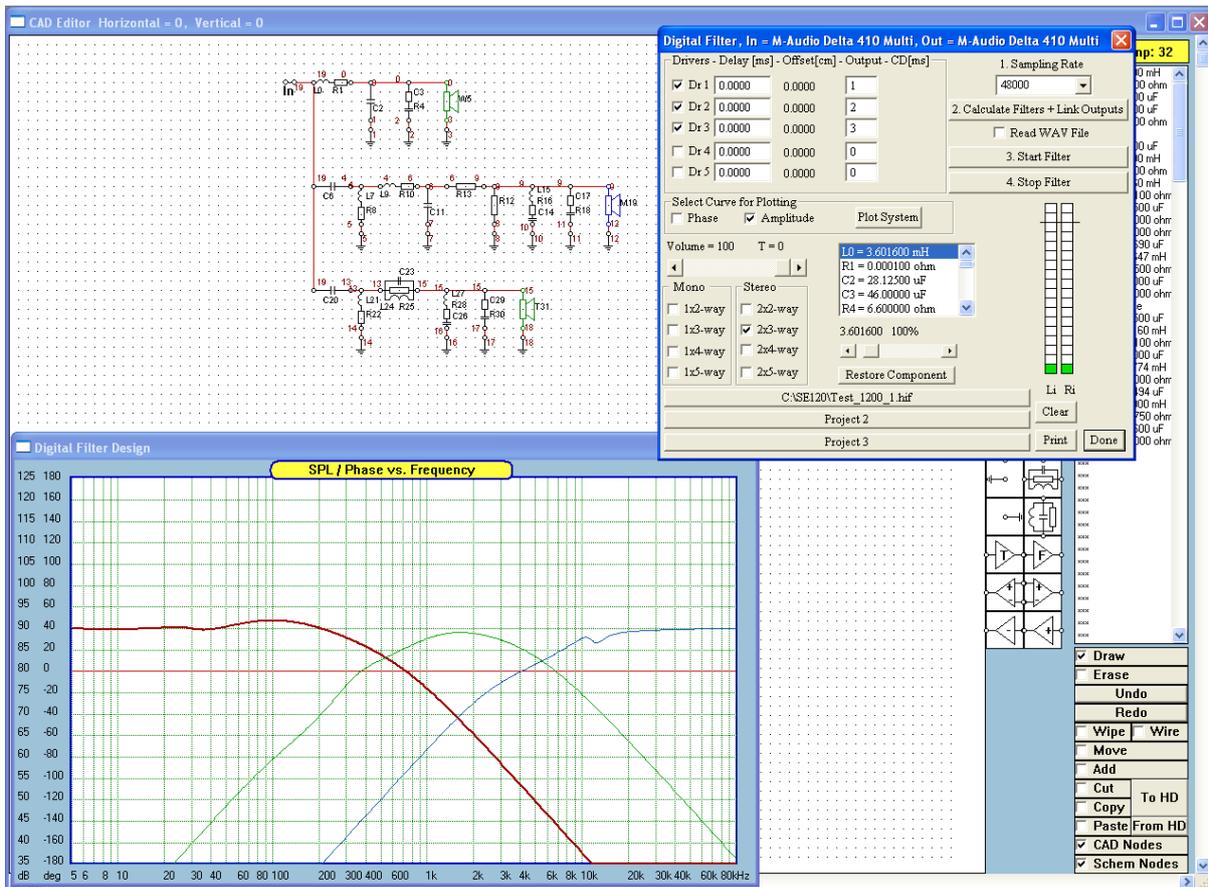
Figure 9.30. Example of a 3-way crossover implemented as digital filters using your sound card.

9.23

You can add **delay to the signal path** for each driver be entering delay in milliseconds in the provided data entry fields. Corresponding physical offsets will be displayed in the "Offset" column.

The technique implemented in this release of the program **does require Microsoft ™ Multi-Media Extensions (MME) to be implemented in your sound card**. This specific approach was taken, because it allows you to use quite inexpensive sound card to its full potential. You can have up to 8 independent DSP filtered outputs, that allow you to create and audition 4-way, stereo speaker system. Certainly, there are many 5-way systems available on the market, however, the vast majority of the designs revolve around 3 or 4-way implementations. One modelling option opened for you is to use Digital Filter function to mimic an active crossover. In this case, in the final implementation, all the drivers would be connected to their individual amplifiers, therefore, the input impedance of the driver would not enter the design equation at all.

There is no need to use loudspeaker symbols anywhere on the schematic for this option. In another option – as shown in the passive crossover example shown above - the magnitude response of each filter is influenced by the actual input impedance of the driver connected at the output of the filter. Here, we are trying to design a passive crossover, that will eventually be used with real drivers connected across designated outputs. Filter **DSP transfer function automatically excludes driver's SPL**, regardless of the selection in the list box on the frequency response screen. However, you need to connect loudspeakers symbols across your crossover output to get the drivers' input impedances included in the Digital Filter transfer function. This will mimic the behaviour of the passive crossover loaded with real drivers, even though for the purpose of the DSP simulation, your drivers are now connected to the soundcards+amplifiers with near-zero output impedances.

When using the Digital Filter option to audition your crossover design, it is unlikely, that your ears will outperform the results of rigorously performed SPL measurements. Particularly, when your crossover has already been optimised for a specific design goal. Please remember, that one of the most critical tasks of the crossover is to protect your drivers from signals they were not designed to handle. Excessive modifications performed on the crossover "by ear", may affect critical system parameters such as: polar response and system power handling. It is therefore strongly recommended, that you proceed very slowly, with full understanding of "pros" and "cons" of what you are doing. It is impossible for Bodzio Software Pty. Ltd. to predict the exact level of compatibility of all hardware installed in your computer. With our best intentions, we suggest to you to take EVERY precaution when operating the digital Filter option.

BEFORE using the Digital Filter option, please consult Chapter 18 – "Computer Related Issues" section for setting your mixer device and general comments on compatibility.

Digital Equalizer

The idea of Digital Filter can be extended into **much more advanced function of Digital Equalization**. Our Digital Equalizer implements dual functionality:

1. Filtering signals intended for a given driver – thus operating as normal crossover section,
2. Equalizing driver's irregular SPL within the filter's bandpass frequency range.

Mathematically, the equalizer performs the following operation:

$$EQ(f) = \frac{a + jb}{c + jd}$$

Where (a+jb) is the driver's transfer function and (c+jd) is the target filter transfer function. The driver's transfer function can be imported into the program from an external measurement equipment or measured by built-in MLS or analogue measurement systems.

The issue of Tails

Please refer to Figure 9.31. In every frequency range where the target SPL curve is ABOVE the driver's SPL curve the equalizer (or the EQ(f) function) will attempt to produce gain to compensate for the driver's faster roll-off. This issue is particularly evident for woofers below the –3dB cut-off frequency and tweeters above the –3dB cut-off frequency. An example of a **severe** "tail issue" (EQ(f) - green curve) is shown below.

Figure 9.31. Unmodified driver's SPL response causes severe tails below 26Hz and above 4kHz.

Here, the EQ(f) will produce rapidly rising frequency response, inversely proportional to driver's roll-off – this is often called "tails issue". **Please note, that wherever the target (blue) curve is ABOVE the driver's SPL (black) curve, the <u>EQ(f) (thick green) curve shows gain above 90.0dB (which is the reference efficiency).</u>** There are three ways to compensate for the tails:

1. Manipulate the nominator the EQ(f) function - here you would employ the Hilbert-Bode Transform to flatten the lowest end of the SPL curve for the woofer.
2. Manipulate the denominator of the EQ(f) function – here, you would modify the frequency response of the target filter.
3. Use (1) and (2) together.

For example, woofer driver can have it's SPL curve modified at the low-end of the audio spectrum. The tool to accomplish this is the Hilbert-Bode Transform. The HB transform will generate mathematically correct transfer function (a + jb) necessary for the rational function EQ(f). It needs to be emphasized, that changing the amplitude alone will not accomplish the task. You need both: real AND imaginary component of the driver's transfer function – this can be easily accomplish using the HB Transform.
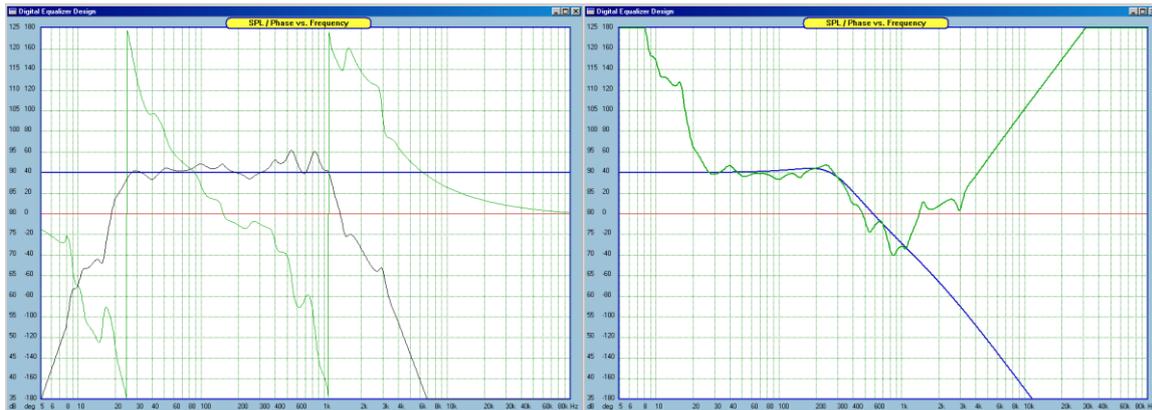
Fig. 9.32. Unmodified driver's response (left) causes severe tails (right) below 26Hz and above 4kHz

Fig 9.33. Modified driver (left) does not create tails (right)

Fig 9.34. Driver 1 HBT settings for the "tail removal"

Recommended Use of the Equalizer

It is assumed, that the equalizer will be used with several audio power amplifiers connected to the output of the sound card. Therefore, input impedance of the drivers will not enter the design process. Also, there is no need for any impedance compensation networks. The Digital Equalizer is invoked from the main menu by selecting the "Digital Equalizer" option.

1. Select your drivers, paying particular attention to acoustic roll-off on low-end and high-end of the operating frequency range.
2. Choose crossover configuration. Here, you should consider the acoustic roll-off speed of the driver vs. the chosen filter. If the target filter rolls-off faster than driver, the tails will not be a problem, otherwise, you will need to modify driver's transfer function (HBT) to get the tails right.
3. Prepare drivers for EQ(f) function using HB transform. All HB Transform parameters are saved into the data file. Ultimately, you will be using project files for the equalizer function and your crossover will be saved as the project file. Note, that you can also use driver file with the equalizer function.
4. Consider how to approach midrange drivers as explained in (2).
5. Open the CAD screen and choose crossover from the built-in selection. It is recommended that the crossover alone be displayed on the CAD screen – no driver impedance modification components should be included in the schematic. However, the crossover can be of any type and include any SPL modification elements. This way you can define quite "esoteric" target functions for all drivers using their crossover sections. **Terminate the crossover with appropriate driver icons**.
6. Inspect the plots and if you are happy with the tails, simply select "Start EQ" button on the "Digital Equalizer" dialogue box. This will start the filtering process and produce the sound at the output of the sound card.
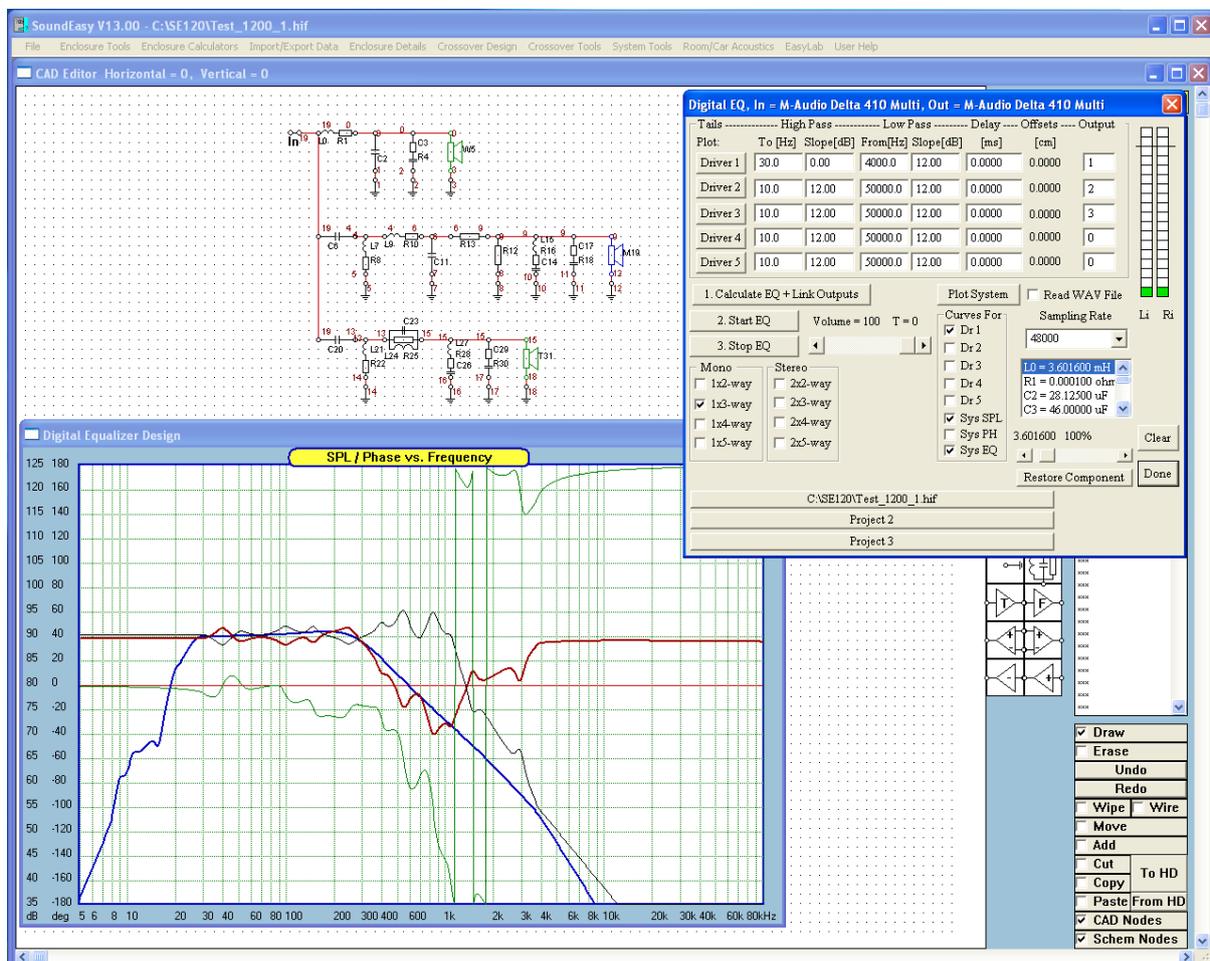

Fig 9.35. Example of all tools used for Digital Equalization

9.27

As for the Digital Filter, before you can play music through the sound card, you need to have the crossover transfer function calculated. To start equalization process, simply press **"Start EQ"** button and to end the process, press the **"Stop EQ"** button. If your sound card is suitable for this function, there will be no warning messages and the sound will appear at the output of the sound card. Here is the functionality of all controls:

1. **Tails Group**
   Driver 1 / Driver 2 / Driver 3 / Driver 4 / Driver 5 - To / Slope / From / To / Delay – Use these data fields to engage Hilbert-Bode Transform to modify Driver 1 SPL curve for the purpose of Tail Correction.

1. **Curves For Group**
   - **Dr1 / Dr2 / Dr3 / Dr4 / Dr5** – Check box to turn ON driver connected to the first pair of nodes in the schematic.
   - **Sys SPL** – Check box to enable plotting of SPL curves.
   - **Sys PH** – Check box to enable plotting of phase curves.
   - **Sys EQ** – Check box to enable plotting of EQ curves.

2. **Calculate EQ + Link Outputs** – Press this button to calculate and plot SPL /Phase of your individual filters (channels) in the crossover. You must have your Driver 1- Driver 5 filter transfer function calculated first, before you can start filtering the music. Every time you change the assignment of crossover outputs to the outputs of the sound card, you MUST activate the new links by pressing this button.
3. **Start EQ** - Press this button to start Digital Equalizer. You should now hear the sound in each channel, filtered by your crossover.
4. **Stop EQ** – Press this button to suspend the Digital Equalizer function.
5. **Plot System** – Press this button to plot total crossover response.
6. **Clear Plots** – Press this button to clear plots.
7. **Print** – Prints the screen.
8. **Export EQ Curve** - This function exports Equalizer SPL curve for a single driver (typically the default driver). This curve can be used as a target for approximate optimisation of active or passive circuits.
9. **Done** – Press this button to close the dialogue box and exit.
10. **Volume** – Use this slider to adjust sound card output.
11. **Mono Group - 1x2 way to 1x4 way** – Check to set the sound card to required mono resolution.
12. **Stereo Group - 2x2 way to 2x4 way** - Check to set the sound card to required stereo resolution.
13. **Sampling Rate** – Select 48kHz OR 44.1kHz ONLY.
14. **Component's List Box** – Double-click on the component you need to adjust.
15. **Component Adjust Slider** – Use this slider to change component's value from 1% - 1000%.
16. **Restore Component** – You can restore the original component's value by pressing this button.
17. **Project 1** – Press this button to make Project 1 current and active.
18. **Project 2** – Press this button to make Project 2 current and active.
19. **Project 3** – Press this button to make Project 3 current and active.

You can add **Delay to the signal path** for each driver be entering delay in milliseconds in the provided data entry fields. Corresponding physical offsets will be displayed in the "Offset" column. You can also change the component value using the slider, while the DF is active. Switches Dr1 – Dr5 will turn ON/OFF drivers and the processed signal. When you start the Digital Equalizer, it is most preferred, that you limit yourself to changing component values only. This is to minimize sound breaks introduced by forcing your computer to deal with additional CPU tasks.

You may also notice, that Digital Filter plotting window can not be customized – for the same reason. If you need to change the size or location of the plotting window, please Stop the filter first, close the control box and select "Digital Filter Plotting" main menu option. Now, you can change the parameters of the window, and they will be "remembered" when you use Digital Filter next time.

9.28

If you need to attenuate a driver by certain amount of dBs, the way to implement this is to attenuate the target function by the required amount OR reduce the gain in the power amplifier by the required amount of dBs. If you run a comparison between several equalizers, you may prefer to use the method shown on Figure 9.36, as

this will automate the whole process significantly and you will not have to change the power amplifier gain every time. In the example shown on Figure 9.36, the L-Pad circuit resistors are Rs = 3.5ohm and Rp = 10.2ohm. These values are calculated by the built-in L-Pad calculator for the required 5dB attenuation and load resistance of 8ohm.
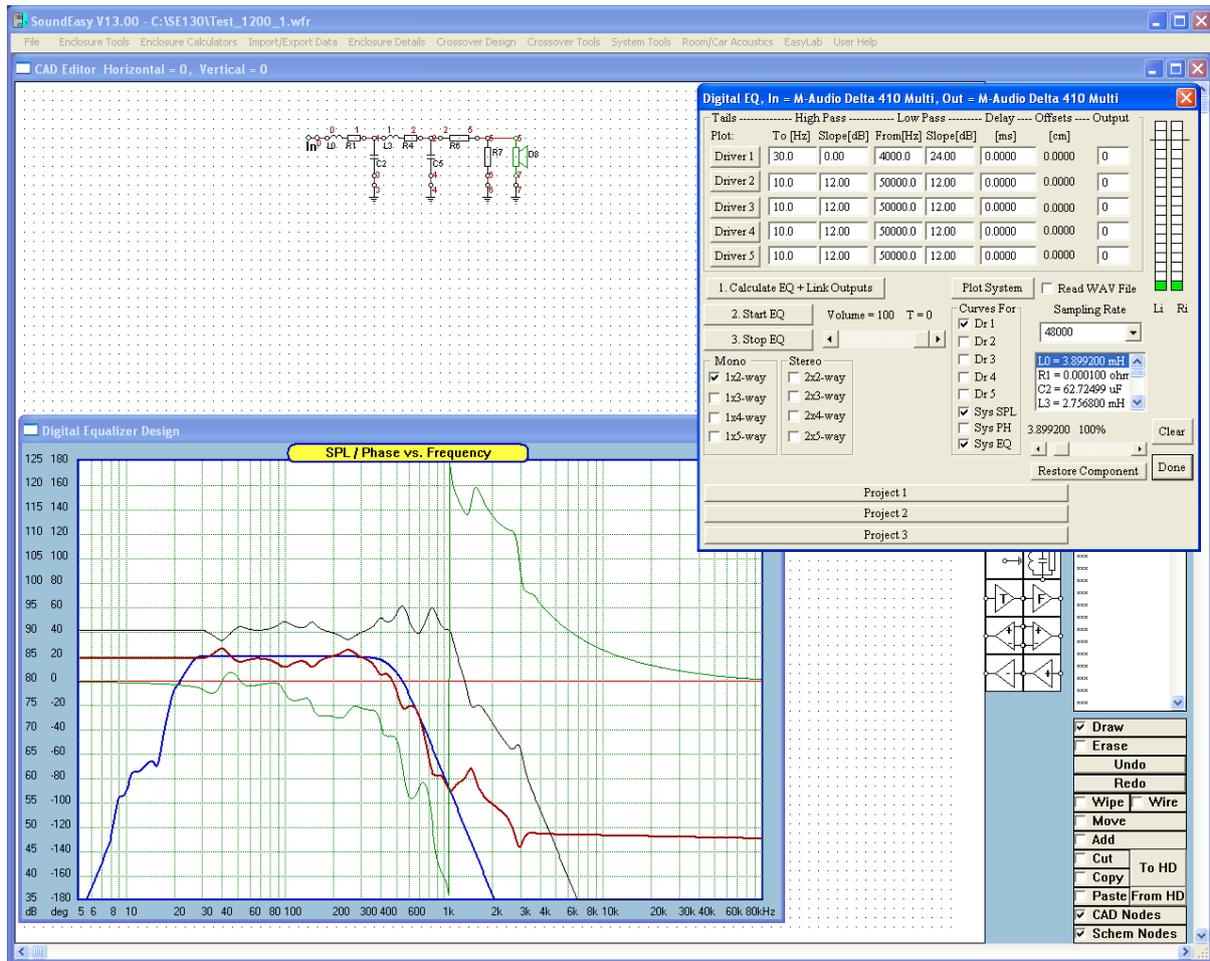
Fig 9.36. Reducing Target level by-5dB – inserting L-Pad 3.5/10.2ohm.

Please remember, that loudspeakers are treated by the equalizer as purely resistive element having a value of R(load) as entered from the Built-in Filter or Crossover dialogue boxes. **All crossover elements (your target curve) are calculated using the same R(load) for terminating resistor. This is necessary for the target functions (ideal filters) to maintain their "perfect" shape, rather than being distorted by the loudspeaker input impedance.** In fact, target function shape can be modified in various ways to produce the desired curve. You can insert any circuit for creating peaks, dips and shelves on the ideal filter frequency response. You can also use active components to generate the target function – however, you MUST be very careful with the gain of the active circuits. Introducing gain in the equalizer may drive the output DAC of the sound card into saturation and clipping

Redirecting Outputs

When developing schematics for different projects, you may end up with driver icons appearing in different order on the schematic. For instance, in **Project 1**, woofer is the first driver (say, Dr1), midrange is the second driver (say, Dr7), and tweeter is the third driver (say Dr13) appearing on the component list. This would automatically assign soundcard Output1 = woofer, Output2 = midrange and Output3 = tweeter.

However, in another **Project 2**, that you want to audition for comparison, the drivers may appear in different order on the schematic, say for instance like this: midrange is the first driver (say, Dr1), tweeter is the second driver (say, Dr7), and woofer is the third driver (say Dr13) appearing on the component list. This would automatically assign soundcard Output1 = midrange, Output2 = tweeter and Output3 = woofer.

Danger !!. You will not be able to simply audition this project on the previously connected amplifier setup. You would have to swap connections from the soundcard to all amplifiers (drivers).

To help you in a situation like this, SoundEaasy allows you to re-direct crossover channels to a specific pair of soundcard outputs. Before we proceed with the explanation of the new feature, it is essential, that you test this feature on your soundcard, and convince yourself, that it operates as intended BEFORE you start using it.

There is a number of combinations possible when performing comparative testing on speaker systems. You can compare mono systems (1x2-way, 1x3-way,1x4-way and 1x5-way) with each other and also stereo systems (2x2-way, 2x3-way, 2x4-way and 2x5-way) with each other.
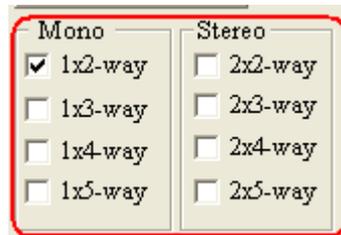
For instance, comparing stereo systems of the same number of filtering channels is fairly straightforward.

1. Connect inputs of your woofer/midrange/tweeter amplifiers to the soundcard outputs, and load **Project 1**. Your DE/DF Control box should be set to "**Stereo 3-way**" system.
2. Enter corresponding numbers into **Output column** of the Control box. Assuming you are working with **Project 1** above, you would enter "**1**" into the top data entry field (woofer to Output 1), then enter "**2**" into the second top data entry field (midrange to Output 2), then enter "**3**" into the third top data entry field (tweeter to Output 3). Enter "**0**"
(**0 = no sound**) into the two remaining fields.

Now, the outputs should sound as follows: Woofer = 1, midrange = 2 and tweeter = 3.

3. Run the DE/DF WITHOUT SPEAKERS CONNECTED, and check, that woofer signal indeed appears on the woofer amplifier output. Midrange signal indeed appears on the midrange amplifier output, and tweeter signal indeed appears on the tweeter amplifier output. Also, confirm, that no other soundcard output emits any signals.

4. For the sake of completeness of your testing, you may try to change numbers in the "Output" column to re-direct signals to different amplifier to make sure, that this scheme works on your sound card. <u>After you change numbers in the "**Output**" column, please do not forget to press "**Calculate EQ and Link Outputs**" button.</u> This will actually change the outputs to the new numbering.

5. Assuming everything works as intended, SAVE the project, and this will also store the Control box settings for **Project 1**.

6. Load **Project 2**.

7. Now, you would enter "**2**" into the top data entry field – this will direct midrange to Output 2

8. Then enter "**3**" into the second top data entry field – this will direct tweeter to Output 3

9. Then enter "**1**" into the third top data entry field – this will direct woofer to Output 1

10. Enter "**0**" (**0 = no sound**) into the two remaining fields.

11. SAVE the project, and this will also store the Control box settings for **Project 2**.
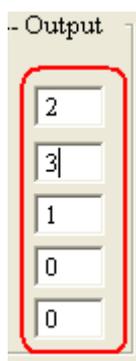
       The above settings should result in identical channel assignment for **Project 1 and Project 2**, and you can now freely swap Projects 1 and 2.

       Comparing stereo systems with different number of filtering channels (say 2-way vs. 3-way system) is nearly identical. **With the exception, that your 2-way project is "checked" in the Control box as "Stereo, 3-way" system.** This way, you are getting access to the same number of soundcard channels (3 pairs of 2 channels that is) as for the 3-way system.

       Assume, your 2-way project is **Project 3**, that you want to audition for comparison, and drivers appear in the following order on the schematic: tweeter is the first driver (say, Dr7), and woofer is the second driver (say, Dr13) appearing on the component list.

       Assuming, that your 3-way system is **Project 1**, and your 2-way system is **Project 3**:

11. Load **Project 3**.
12. Now, you would enter "**3**" into the top data entry field – this will direct tweeter to Output 3
13. Then enter "**1**" into the second top data entry field – this will direct woofer to Output 1
14. Then enter "**0**" into the third top data entry field – this will create no sound for the dummy driver.
15. Enter "**0**" (**0 = no sound**) into the two remaining fields.
16. SAVE the project, and this will also store the Control box settings for **Project 3**.

Now, the outputs should sound as follows: Woofer = 1, dummy midrange = 0, no sound, and tweeter = 3.

       The above settings should result in identical channel assignment for **Project 1 and Project 3**, and you can now freely swap projects 1 and 3. Comparing mono systems should follow the same guidelines, except please use "Mono" group of check boxes.

       As one can see from the above comments, comparison between all sorts of systems is quite straightforward. You need to set your control box to the most advanced system for each group of projects, and enter "0" in the "Output" column to mute the dummy outputs for less advanced systems. Probably the most critical part of the above process, is to make sure, that your soundcard + driver are actually working correctly in the above scheme. So, please test thoroughly before you start using it.

In summary, the equalizer creates a mirror-image of the driver's frequency response and modifies it such a way, that the combined transfer function (driver's SPL + EQ(f)) sums to the requested target curve. The target curve may be the "ideal" LR 4th-order filter or quite "esoteric" curve created specifically for R&D purposes.

It is impossible for Bodzio Software Pty. Ltd. to predict the exact level of compatibility of all hardware installed in your computer. With our best intentions, we suggest to you to take EVERY precaution when operating the digital Filter option. BEFORE using the Digital Equalizer option, please consult Chapter 18 – "Computer Related Issues" section for setting your mixer device and general comments on compatibility.

Disclaimer

Bodzio Software Pty. Ltd. assumes no responsibility for any damage to the hardware running the SoundEasy program or any equipment connected to this hardware. It is entirely User's responsibility to check the signals generated from the sound card before connecting to any external equipment.

See system example below. A 3-way, 24dB/oct Butterworth stock crossover was used with cut-off frequencies of 400Hz and 4000Hz.
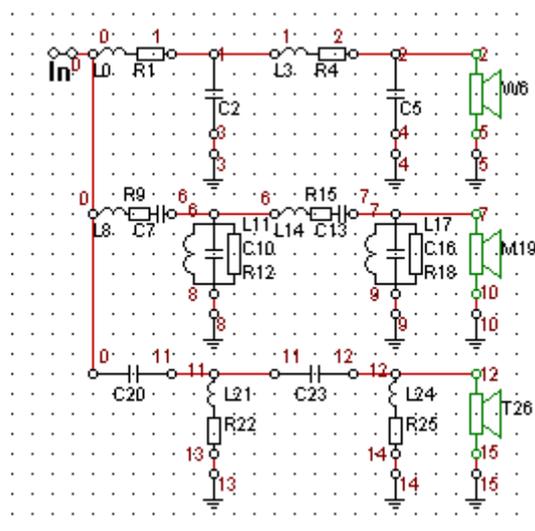
Figure 9.37. 3-Way system ready for Digital EQ function

Figure 9.38. Woofer, midrange and tweeter drivers frequency responses.
9.32
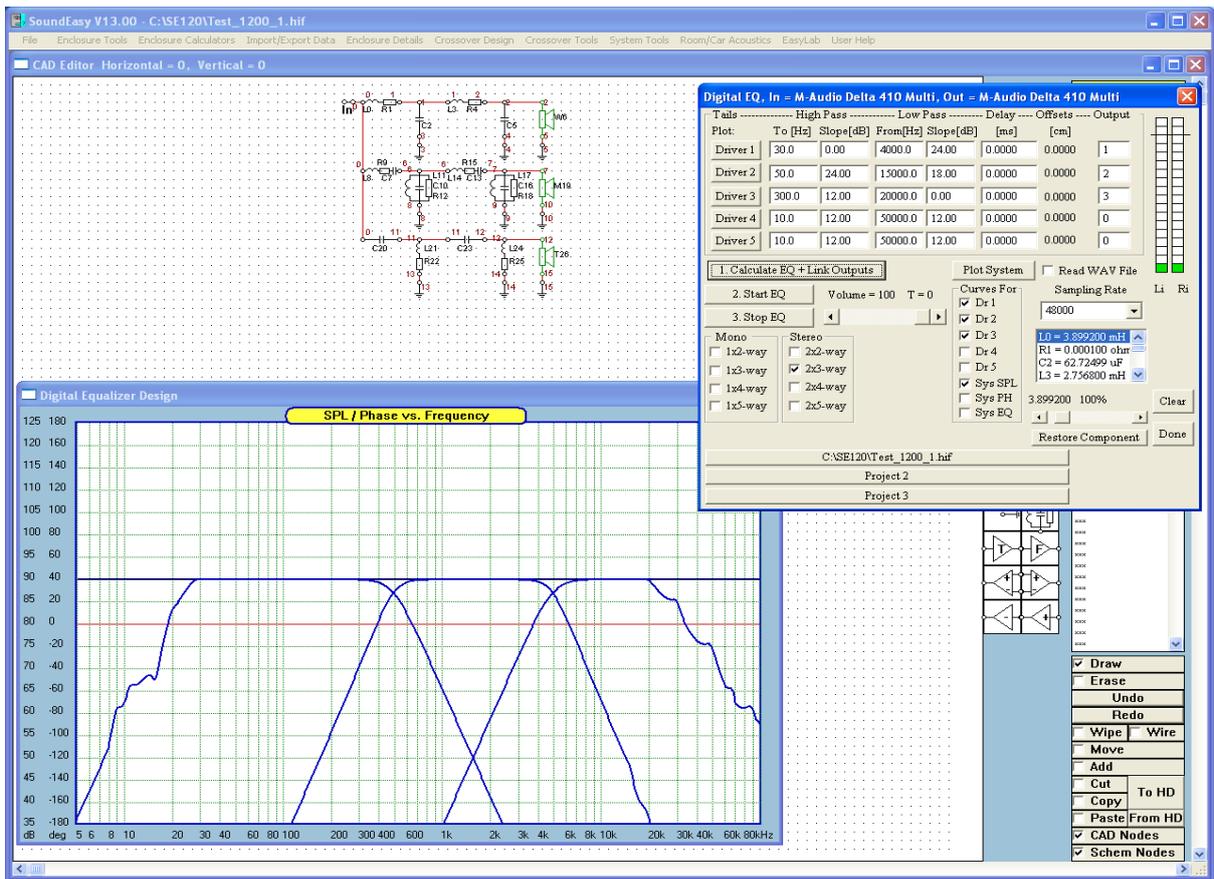
Figure 9.39. All 3 drivers were equalised – these are their system (driver+EQ) responses
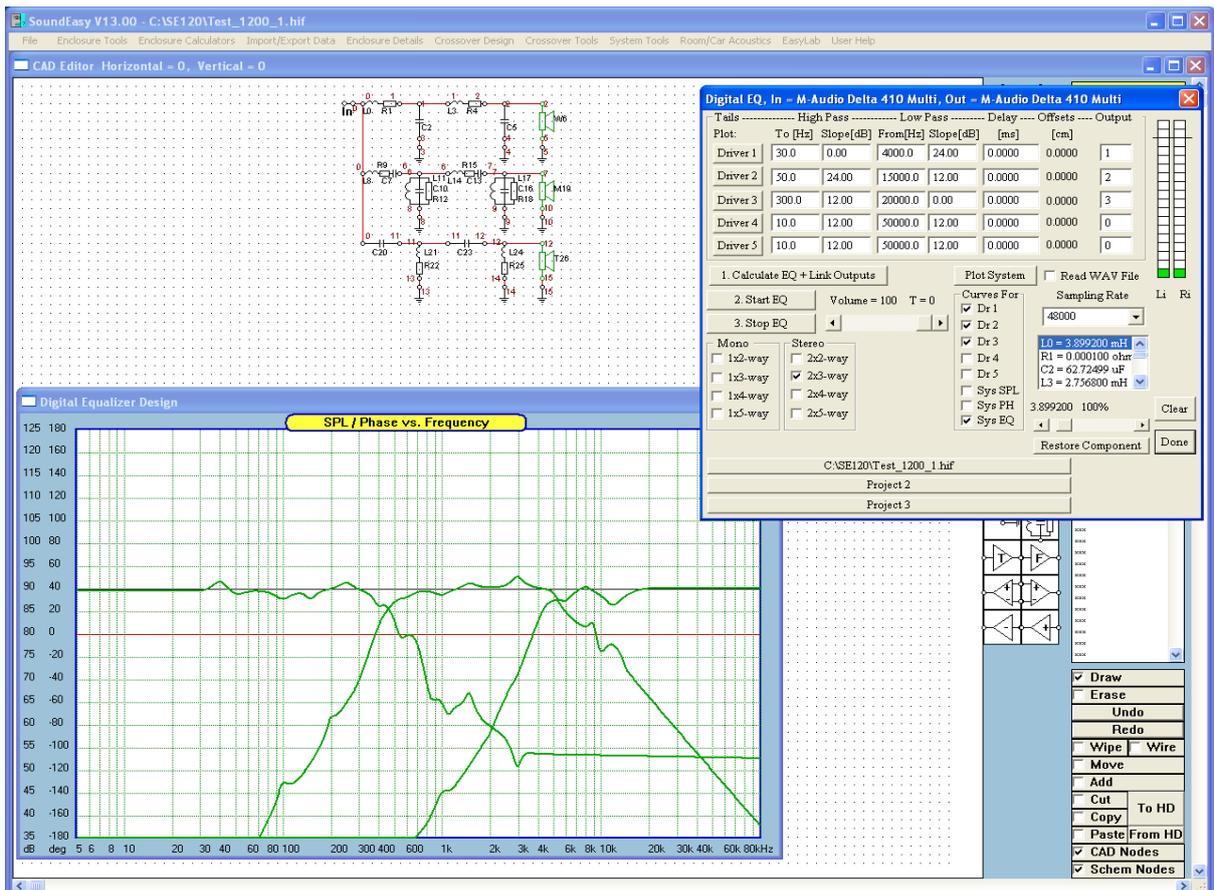
Figure 9.40. All three EQ curves used to obtain responses as on Figure 9.39

9.33

Time Delayed Filters

Time delayed filters are based on the requirement, that low-pass and high-pass channel responses must sum to a unity. $H_L(s) + H_H(s) = 1$

This requirement alone did not unfortunately produced good polar responses and J. Vanderkooy came with a simple modification to the above formula, that showed much greater promise – a time delay element was introduced.

$$H_L(s) + H_H(s) = \exp(-\tau s) \quad \text{OR} \quad H_H(s) = \exp(-\tau s) - H_L(s)$$

At the time when the modification was proposed, those filters could not be realized. There was no readily available time delay elements. Now, with the advent of DSP techniques, those limitations have been lifted and you can implement the whole family of time-delayed filters quite easily. Auditioning of the filters can be facilitated via Digital Filter option.
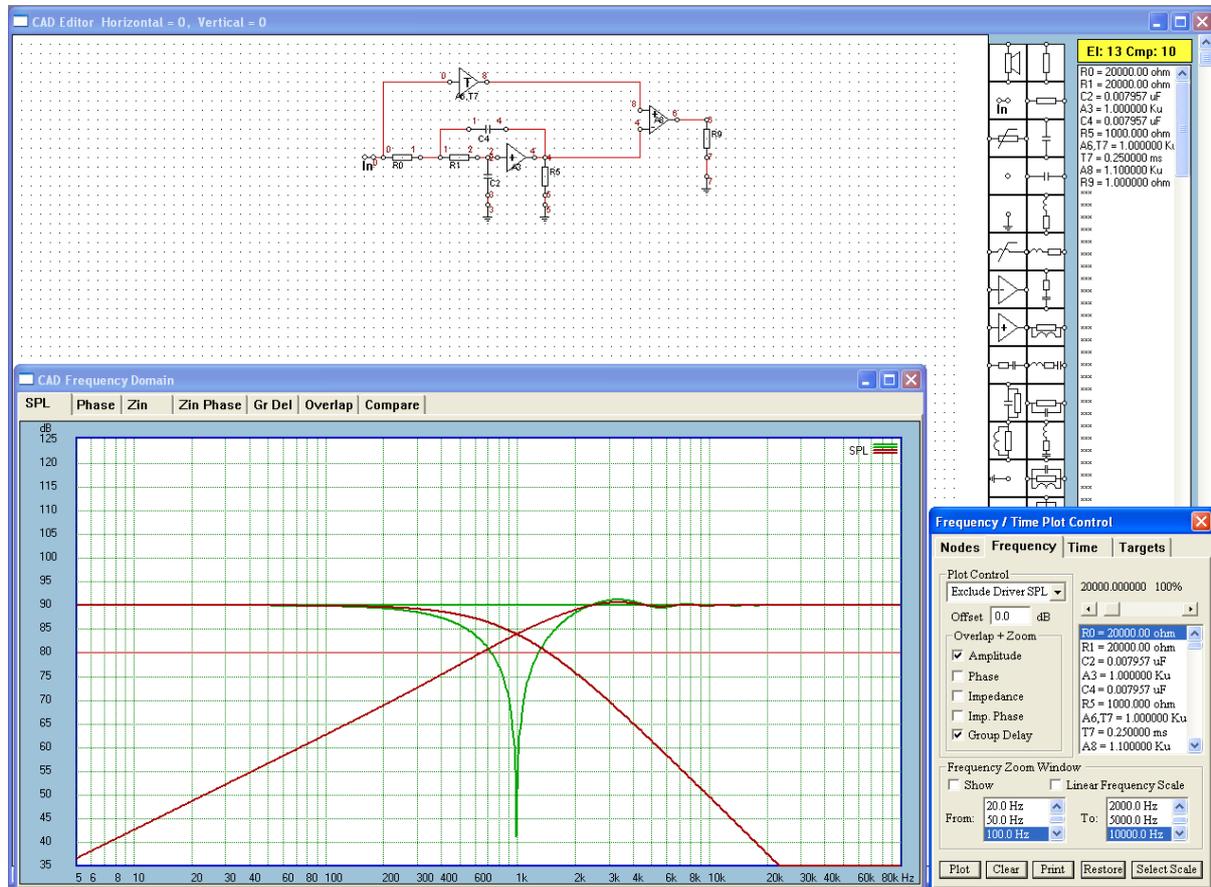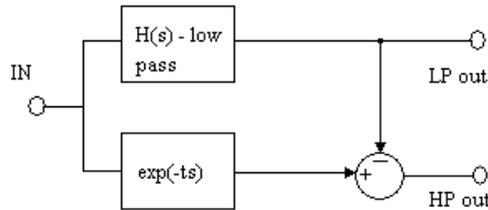
Figure 9.41. 2-way crossover built using time-delayed filters. LP=Node4, HP=Node6. (Note inverted response)

Low-Pass Linkwitz-Riley filter with cut-off frequency of 1000Hz. Delay element introduces 0.250msec delay and is connected to the non-inverting input of the differential amplifier with gain = 1.10. Summed response is "perfectly flat" line at 0.0dB level.

Please note the R9 load resistor for A8 – you must NOT leave the output unconnected.
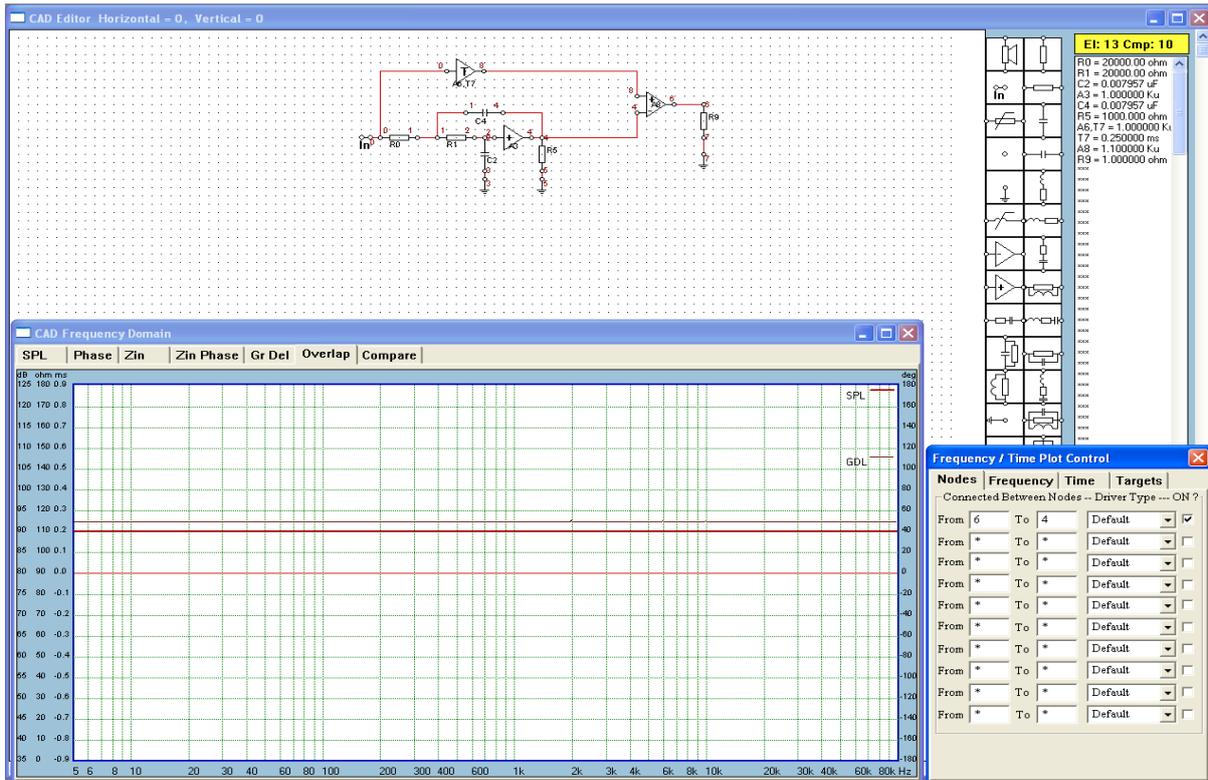
Figure 9.42 Group delay of the time-delayed crossover - CONSTANT

The delay set in the delay element is recommended to equal the low-pass section group delay at 0Hz, that is 0.250ms. Group delay is implemented as the first frequency derivative $G(\varpi)$ of the phase response $P(\omega)$.

$$G(\varpi) = \frac{\partial P(\varpi)}{d\varpi}$$

Since the group delay is constant, this indicates, that the phase response is linear.

Driver Delay Offset using Digital Filter

When implementing Time Delayed filters or crossovers, you may still desire to provide additional delay for some drivers to compensate for the difference on acoustic centre offsets. This delay can be conveniently implemented using the fields provided in the "Digital Filter" dialogue box. The delay must be entered in **milliseconds** as positive numbers.
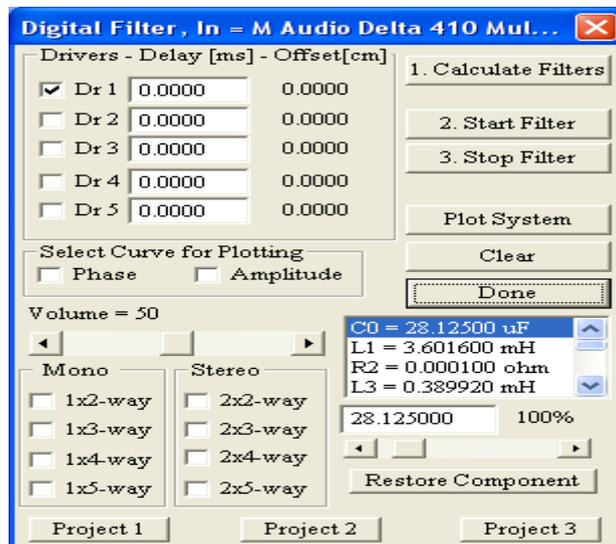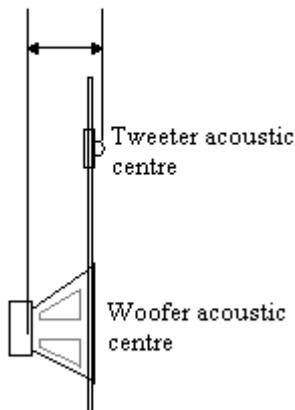
Figure 9.43. Illustration of the acoustic centres difference.
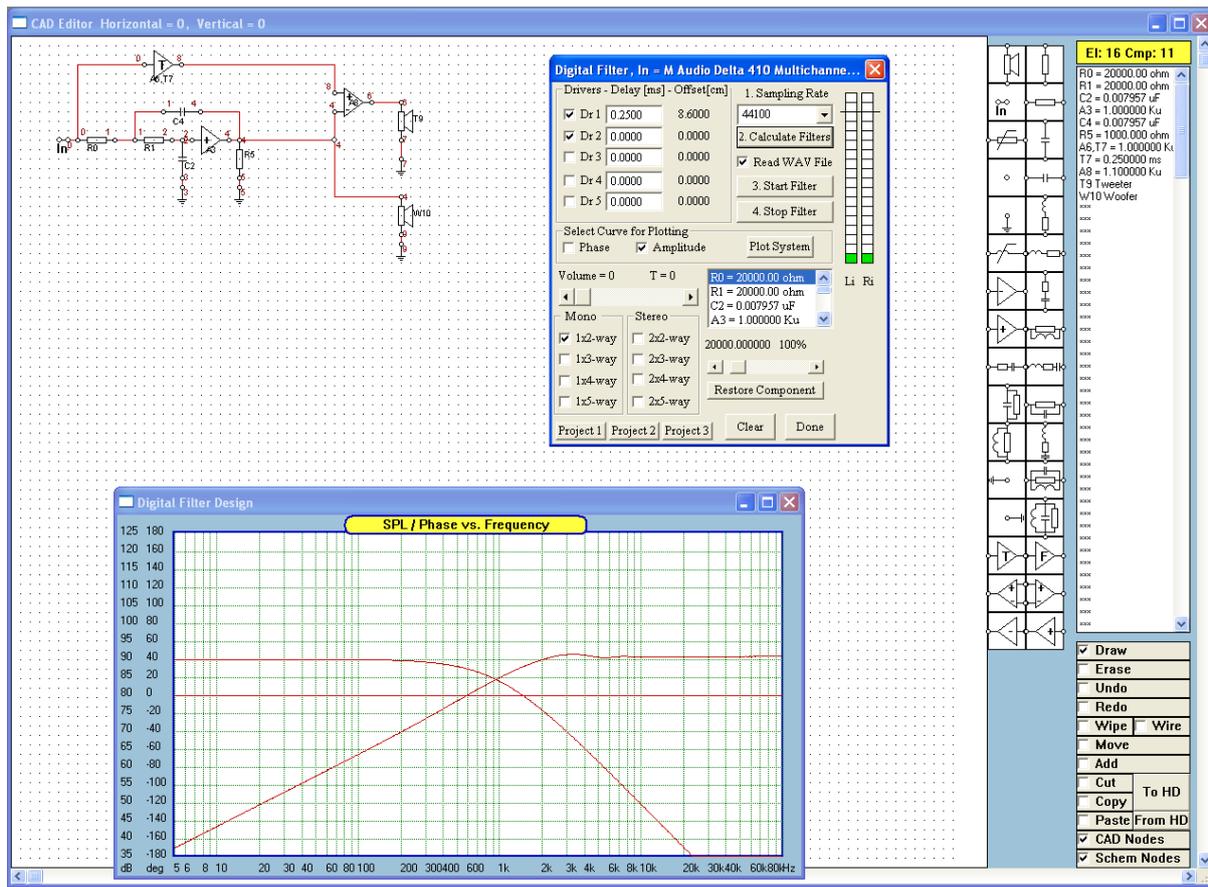
Figure 9.44. Digital Filter with additional delay of 0.25ms for driver 1.

Super-Component

The concept of **Super-Component** introduces a very powerful function into the program. The Super-Component (SC) can be described as follows:

1. Electrical implementation is identical to Voltage-Controlled-Voltage-Source (VCVS), with editable gain, input resistance and output resistance.

2. Frequency response of the SC can be any of the built-in filters or networks. **In addition, the SC can assume user edited frequency response loaded from previously saved target file.**

3. The SC can also introduce a pure time delay into the circuit, and all parameters from (1) are still available.

Practical implementation of the SC would be limited to built-in filters or networks. You would simply substitute the SC with the corresponding electrical network. The network could be actually quite complex, as the SC allows you to use any built-in filter/network configuration. Using the SC as a time delay element or assigning user edited frequency response to it, can be implemented in practice via Digital Filter or Digital Equalizer. Selecting and assigning various frequency responses from the "Edit Super-Component Data" dialogue box is very simple, as the dialogue box works the same as previously described filter/crossover selection dialogue boxes.
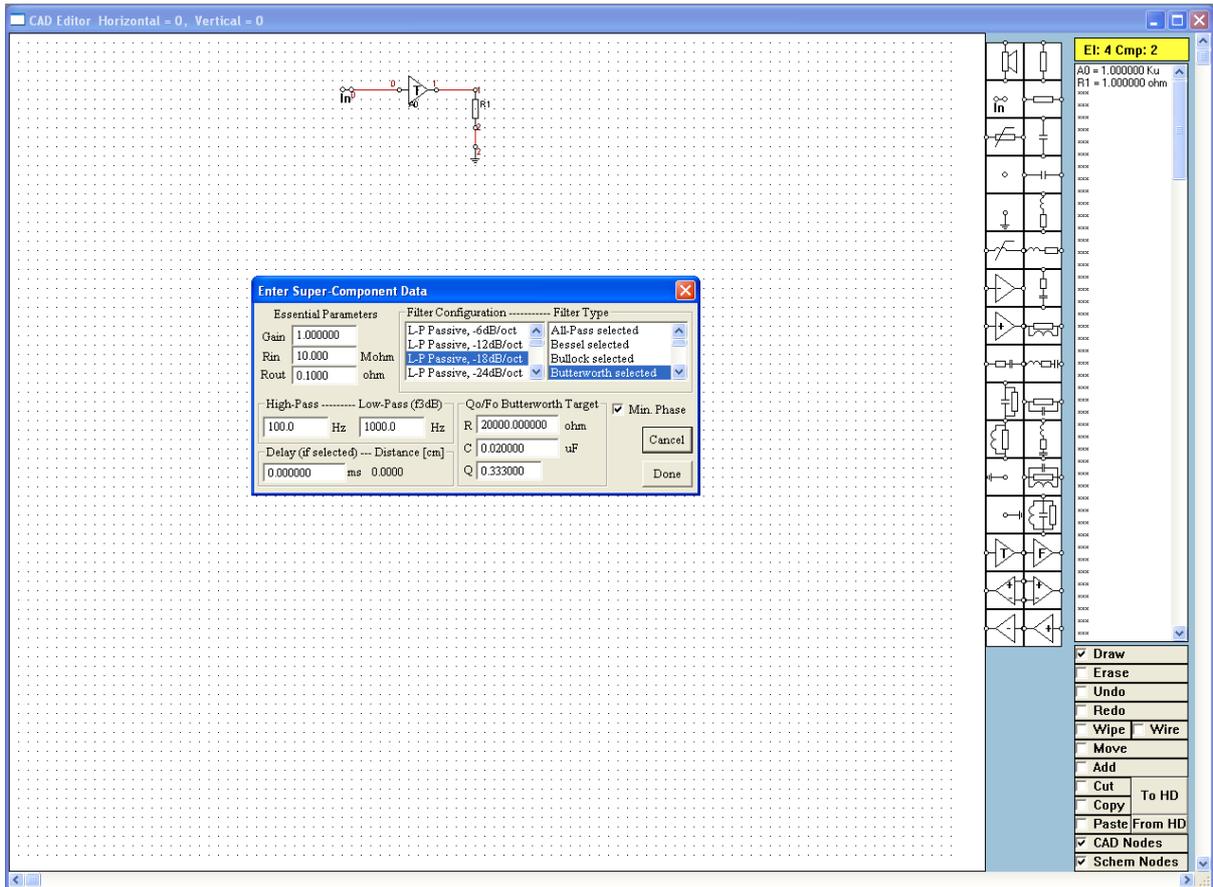
Fig 9.45. Super-Component Editor dialogue box

1. **Essential Parameters Group**
   a. **Gain** – Use this field to enter Super-Component's gain.
   b. **Rin** – Enter input resistance of SC.
   c. **Rout** – Enter output resistance of SC.

2. **Filter Configuration** – Select order of the filter from this list box.

3. **Filter Type** – Select filter's type from this list box

4. **High-Pass** – Enter high-pass 3dB cut-off frequency.

5. **Low-pass** – Enter low-pass 3dB cut-off frequency.

6. **Delay** – Enter required time delay. You will see the equivalent acoustical distance calculated.

7. **Qo/Fo Butterworth Target** – Enter additional filter parameters for certain Butterworth filters.

8. **Cancel / Done** – Use as described on the buttons.

9. **Min. Phase** – See below.
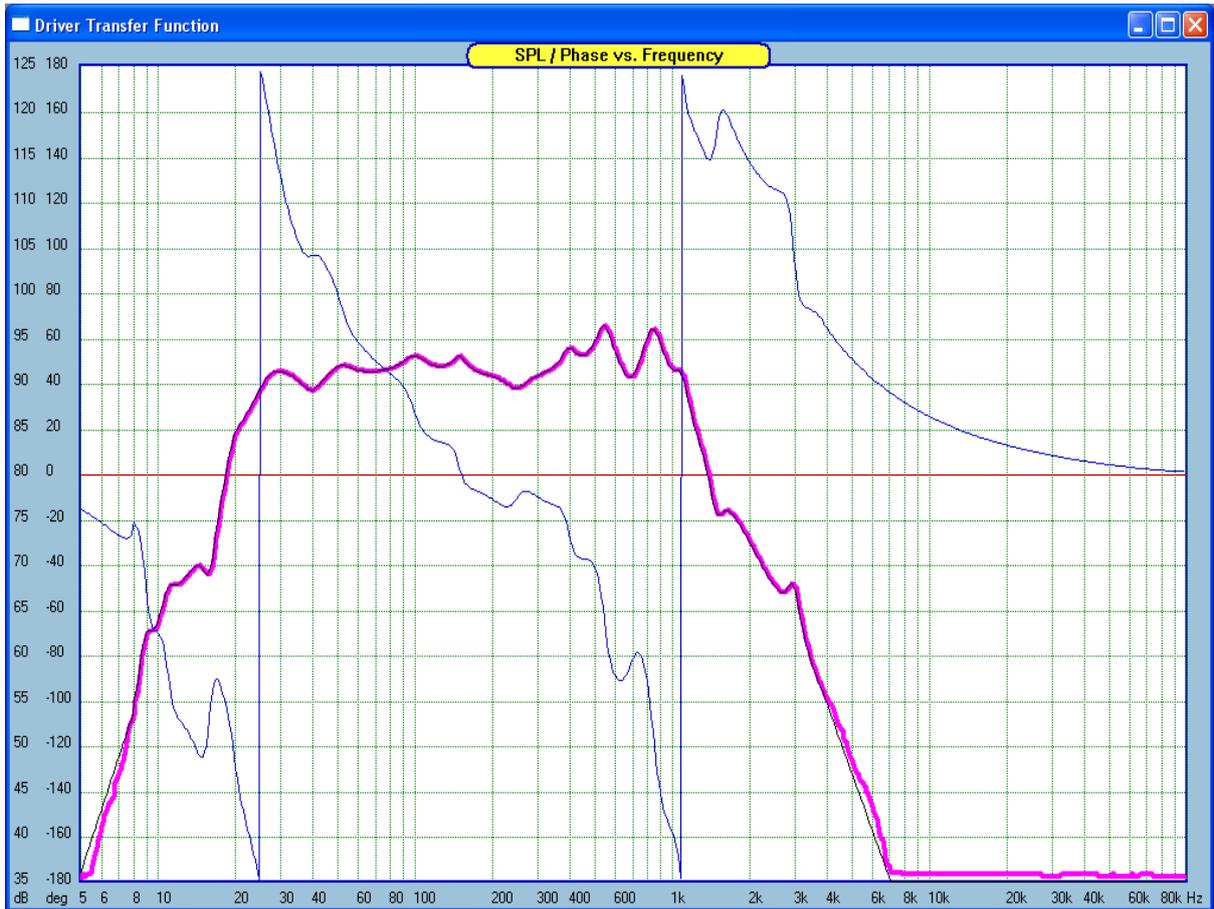
Figure 9.46. **Step 1** – Editing frequency response.
Step 2 – Running HBT on the curve
Step 3 - Saving the results of HBT as a target file.
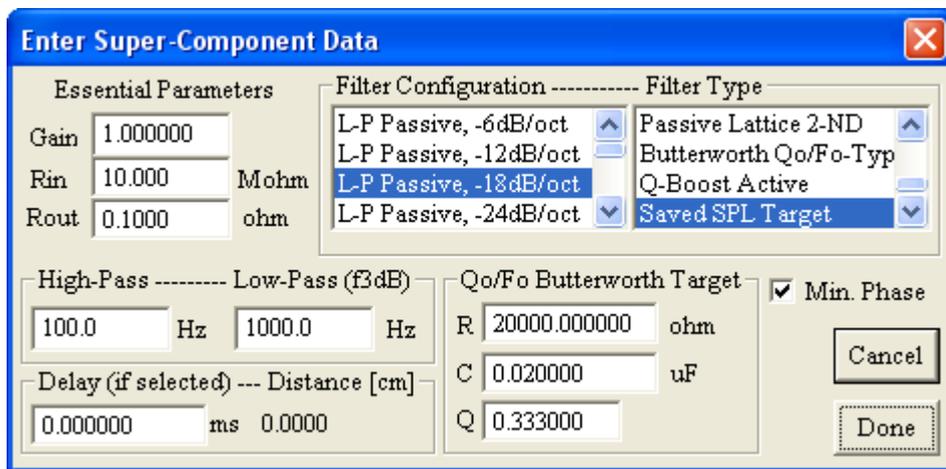
Figure 9.47. **Step 4** - Selecting user edited frequency response into Super-Element
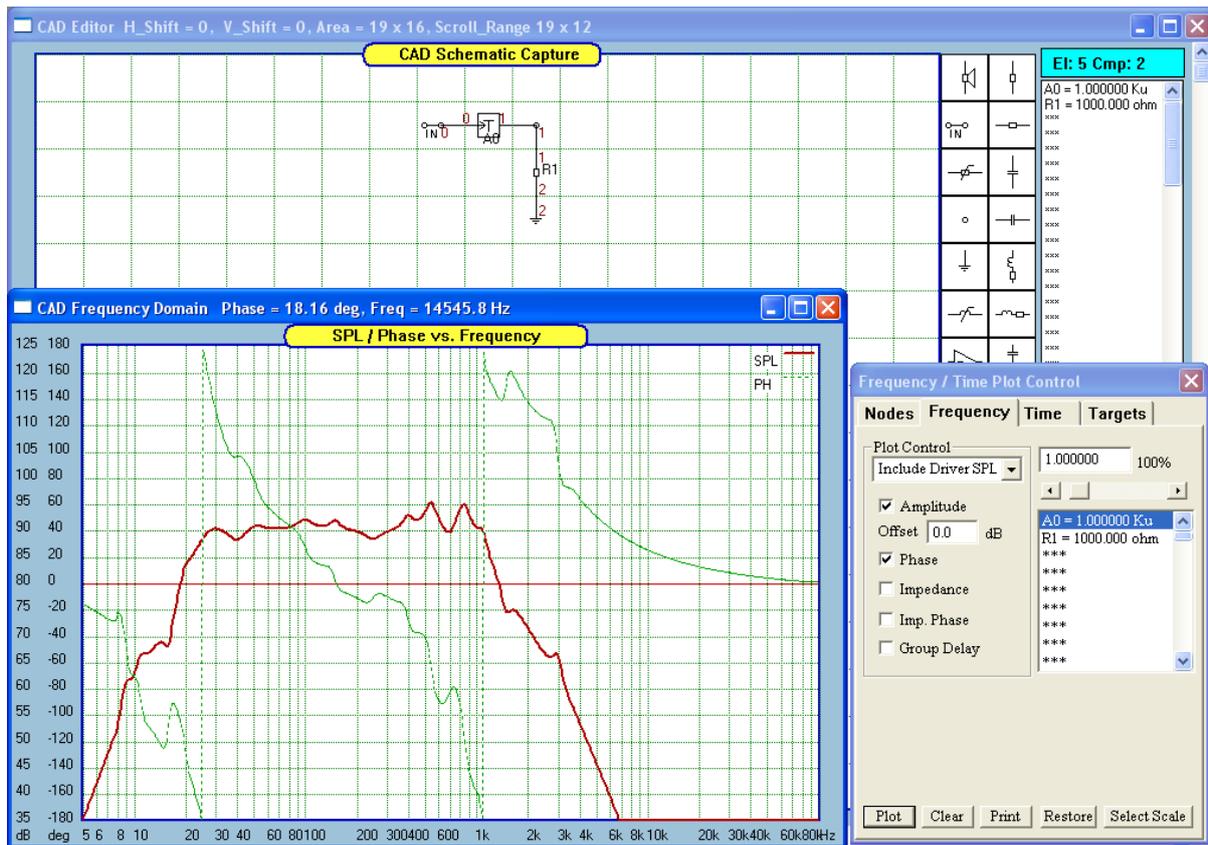
Figure 9.48. Frequency response of the Super-Component is the same as edited on Figure 9.46.

Mega-Components

The frequency response plotted on the "Frequency Domain" CAD plotting screen can be saved into the target file as well – you can only do this when the plotting screen is **opened and active**. This immediately opens another opportunity for the application of the Super-Component. The idea is explained using Figures 9.49 and 9.50.

Figure 9.49 shows an example of 5 Super-Components with assigned very different frequency responses: 1 low-pass filter and 4 band-pass filters. Output impedance of each Super-Component is set to 1ohm and outputs are connected in-parallel, so there will be some mutual loading effect reducing the gain (by 4times) of the whole system in this particular configuration. This is never a problem, as the mutual loading can be compensated by increasing the gain of each Super-Component anyway.

After the frequency response was plotted, it was then saved into the target file. In the next step, the screen was cleared and a simple Super-Component network was created as shown on Figure 9.50. This time, the Super-Component was assigned the frequency response from the previously saved file and gain set to 10x (or 20dB). Now, the single Super-Component exhibits the combined frequency response of the previous 5 Super-Components. This turns it effectively into a **Mega-Component**.

It is easy to observe that the process describe above can be carried out recursively infinite number of times. The resulting frequency response, created this way, can be extremely complex.

In addition, the **Super-Components and Mega-Components can save you drawing space** significantly. As you can see, a circuit, that can be as big as the whole screen can now be substituted in many cases by a single component. As the **Super-Components and Mega-Components** are implemented as VCVS, it makes them particularly suitable for active circuit implementation. However, since the input and output impedances are editable, you can easily use these elements in passive filters and crossovers. Please note, that depending on their complexity, it may not be possible to implement the **Super-Components and Mega-Components** in any other way, except for Digital Filter and Digital Equalizer. This is where they are mostly intended to be used.

9.39
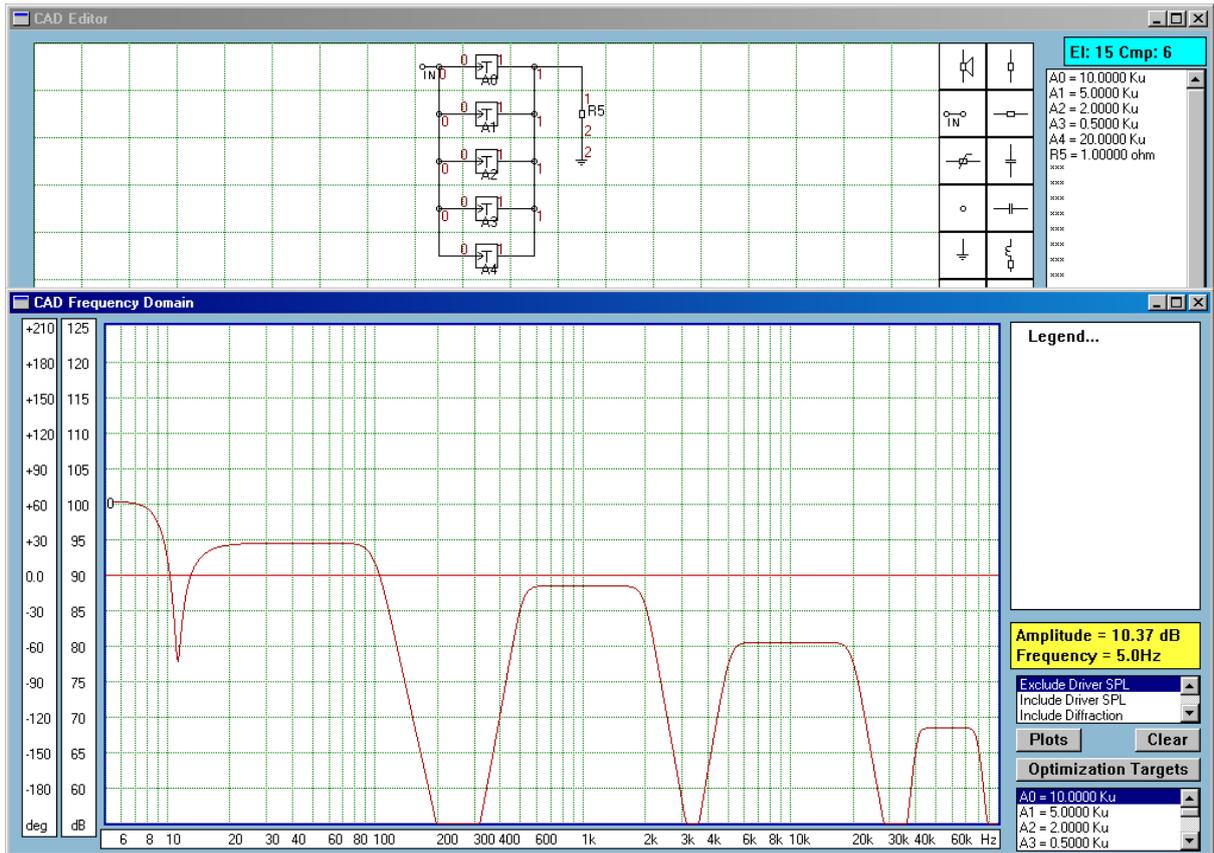
Figure 9.49. SPL of 5 **Super-Components** connected together.
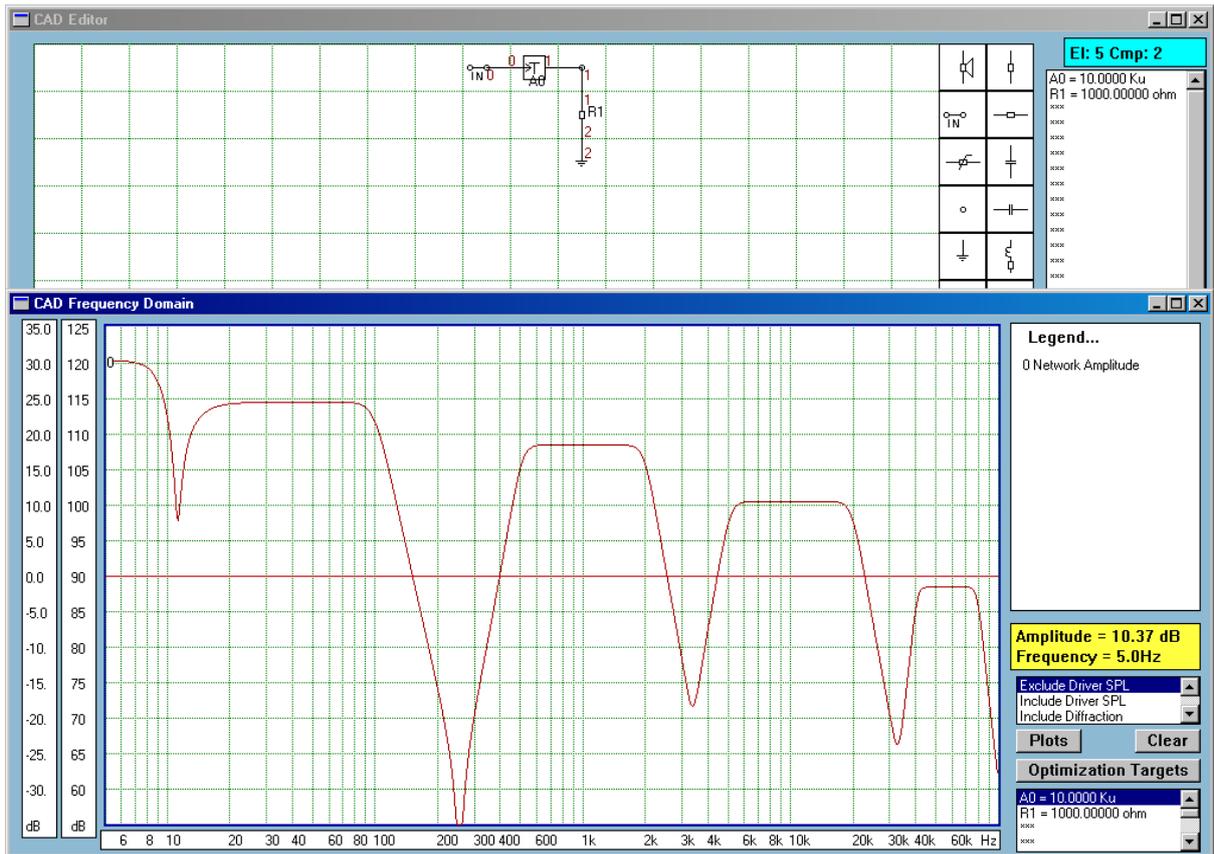
Figure 9.50. Frequency response from Figure 9.49 was saved and assigned to single **Mega-Component**.

Linear Phase Filters

Linear Phase Filters can be created using Super-Component's feature called **"Min.Phase".** When this Option is checked, the phase response of the circuits created via Super-Component is the normal, "minimum-phase" response. However, un-checking the box causes the phase to become linear, irrespectively of the magnitude response. Interestingly, the magnitude-phase responses no longer hold Hilbert-Bode relationship. Phase is controlled via delay and can be set to zero or any constant, positive value by entering data into the **"Delay"** field.
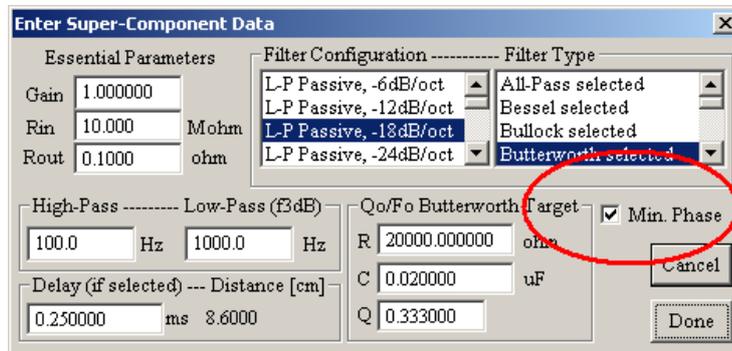
Figure 9.51. Enabling Minimum-Phase response.
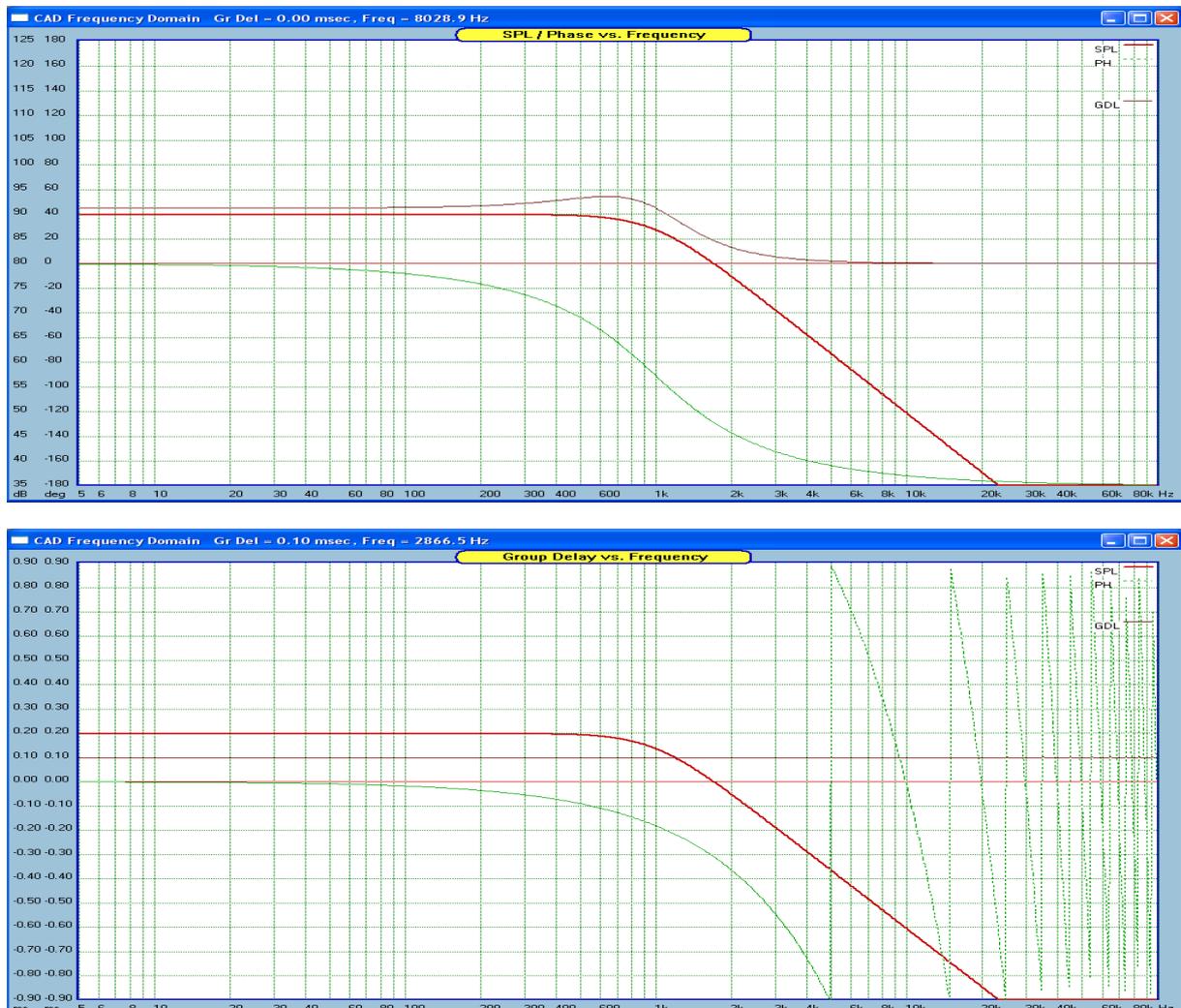
Fig 9.52. Here is an example of low-pass filter with minimum-phase (upper) and linear phase (lower).

Please note the typical group delay in the upper plot and constant group delay (0.10 ms) on the lower plot.

9.41

Modular Crossover

The concept of Modular Crossover (MC) design and optimization is intended to compliment the traditional "discrete component" crossover approach. It is anticipated, that modular crossover will find its use in emulating and selecting the parameters and optimization of DSP Crossover hardware already available on the market. A good example of such device is Behringer DXC2496 device. The DCX2496 is capable of emulating several popular filter configurations such as low-pass, high-pass, shelving filters and variable-Q type filters. The DCX2496 is a very fast, real time filtering device, but lacks SPL optimization functions or SPL importing functions for proper crossover design. The MC approach would allow you to emulate the behaviour of DCX4296 and at the same time, design and optimize the crossover, including desirable driver's AC offsets (delays).

One can hope and anticipate, that future DSP hardware filters will allow for significant increase in emulation capabilities in terms of greater number of Impulse Response coefficients to be processed. This would eventually facilitate implementation of the Digital Equalizer functionality already provided in SoundEasy.

Using the MC approach, crossover filter's **Transfer Function is expressed in terms of cut-off frequency, slope roll-off (or Q-factor) and design type**. This is in contrast to a list of component values for traditional, discrete component-type crossovers. For instance, using MC methodology, we would be talking about a second-order, Butterworth low-pass filter, with a cut-off frequency of 1000Hz.

All MC filters are implemented as building-blocks. Current software implementation offers you a number of filter types and configurations to choose from. You can find the MC icon tagged as **"F" (for Frequency-dependant component)** on the CAD screen component pickup section, together with the other components. When you pick-and-place the F-component on the CAD screen, you can double-click the left-mouse button to open a control dialogue box. This box allows you to select the desired parameters for this particular F-component. The box is shown below. All rules and recommendations for drawing schematic apply to the F-component as well, so they will not be repeated here. Operation of the dialogue box is really self-explanatory.
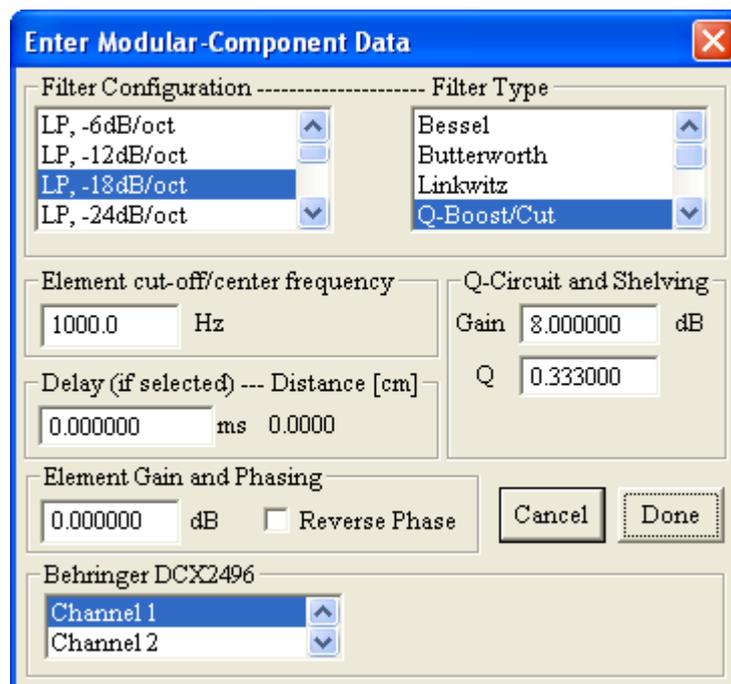
Figure 9.53. F-component control box

You will notice, that each F-component is displayed on the CAD screen with a short list of it's most characteristic parameters. The list occupies some space on the drawing area, therefore it is advisable to space the F-components more sparsely on the screen, to enable all information to be displayed without overlapping.
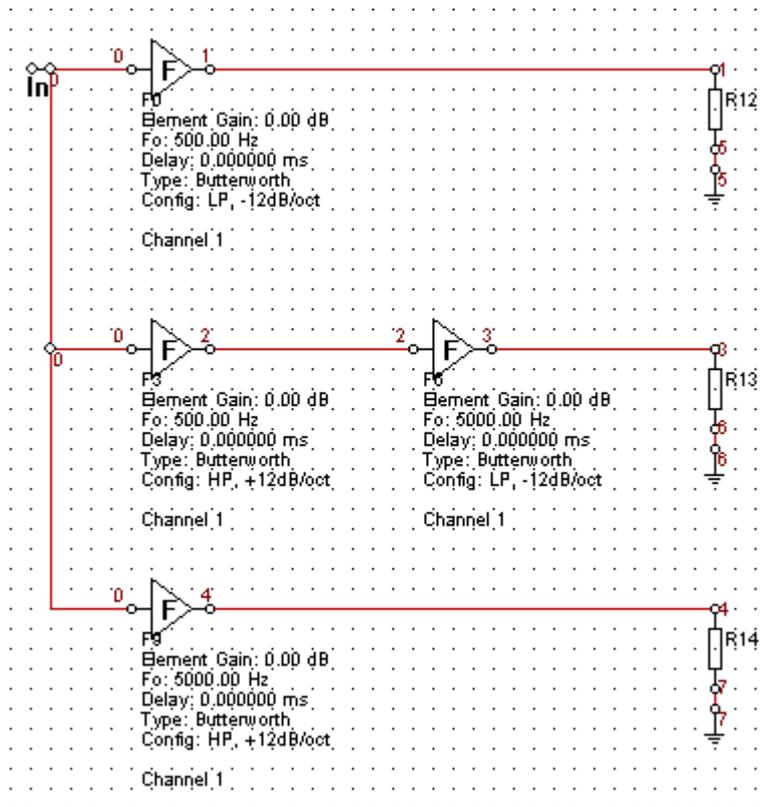
Fig 9.54. 3-way crossover with 500/5000Hz cut-offs. Note F6 has to invert midrange channel.

Current release of the program implements the following F-components:
1. +/-6dB/oct Low-pass shelving filter,
2. +/-6dB/oct High-pass shelving filter,
3. +/-12dB/oct Low-pass shelving filter,
4. +/-12dB/oct High-pass shelving filter,
5. +6, +12, +18, +24, +48dB/oct Butterworth High-pass filters,
6. -6, -12, -18, -24, -48dB/oct Butterworth Low-pass filters,
7. +6, +12, +18, +24, +48 Bessel High-pass filters,
8. -6, -12, -18, -24, -48dB/oct Bessel Low-pass filters,
9. +12,+24 Linkwitz High-pass filters,
10. -12, -24dB/oct Linkwitz Low-pass filters,
11. Q-Peaking/Notch filter (or Parametric)
12. Phase inversion. **The A15 is really redundant, as you can invert phase by checking the "Reverse Phase" box on Fig 9.53.**

All F-components allow for individual gain adjustment. The gain is expressed in dB's, and you can enter negative gain as well for attenuation purposes. **In addition, all F-components can have a time delay added to their Transfer Function.**

The F-components can be mixed with all standard passive (RLCs) and active (OPAMPs) components as you wish. The F-component has a very high input impedance and very low output impedance, so effectively it behaves in the circuit as a VCVS (Voltage Controlled Voltage Source, or an OPAMP if you prefer) with **frequency dependant** Transfer Function – hence it's name the "F-component".

Frequently, the MC approach may speed-up your design. For instance, imagine, that you have your 3-way crossover designed already with discrete components on the CAD screen, but you want to change the cut-off frequency of a low-pass filter. Depending on the complexity of the filter, a number of components would have to be re-calculated and changed. To accomplish this, you could load another low-pass filter onto the CAD screen, with components calculated for the new frequency and re-link the output to the corresponding driver. Then check the system SPL for the desired outcome. If OK, you could erase the old filter (CUT / PASTE function) and use the new one. You could do this again and again until the circuit worked as desired. However, the MC approach would be a better option to start with.
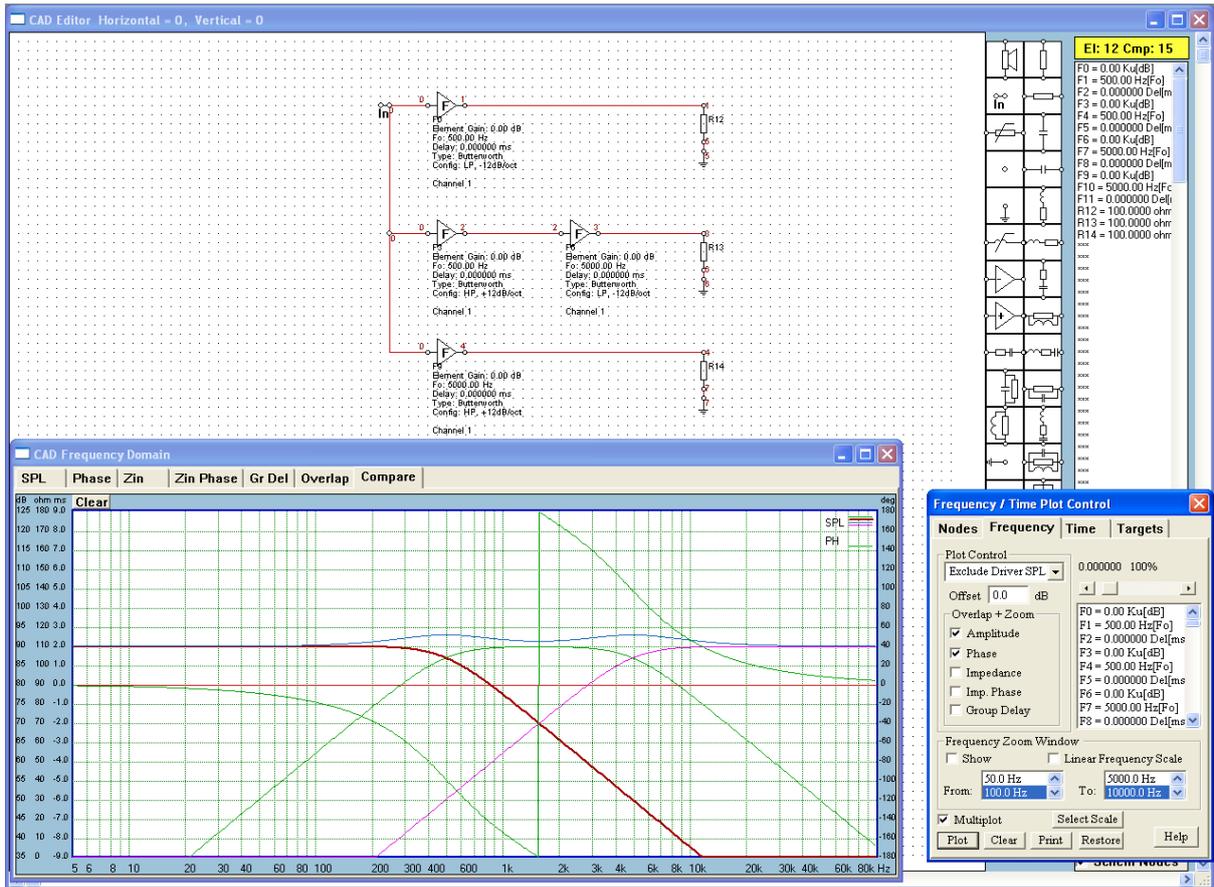
Fig 9.55. An example of a 3-way system crossover with SLP and phase plots.
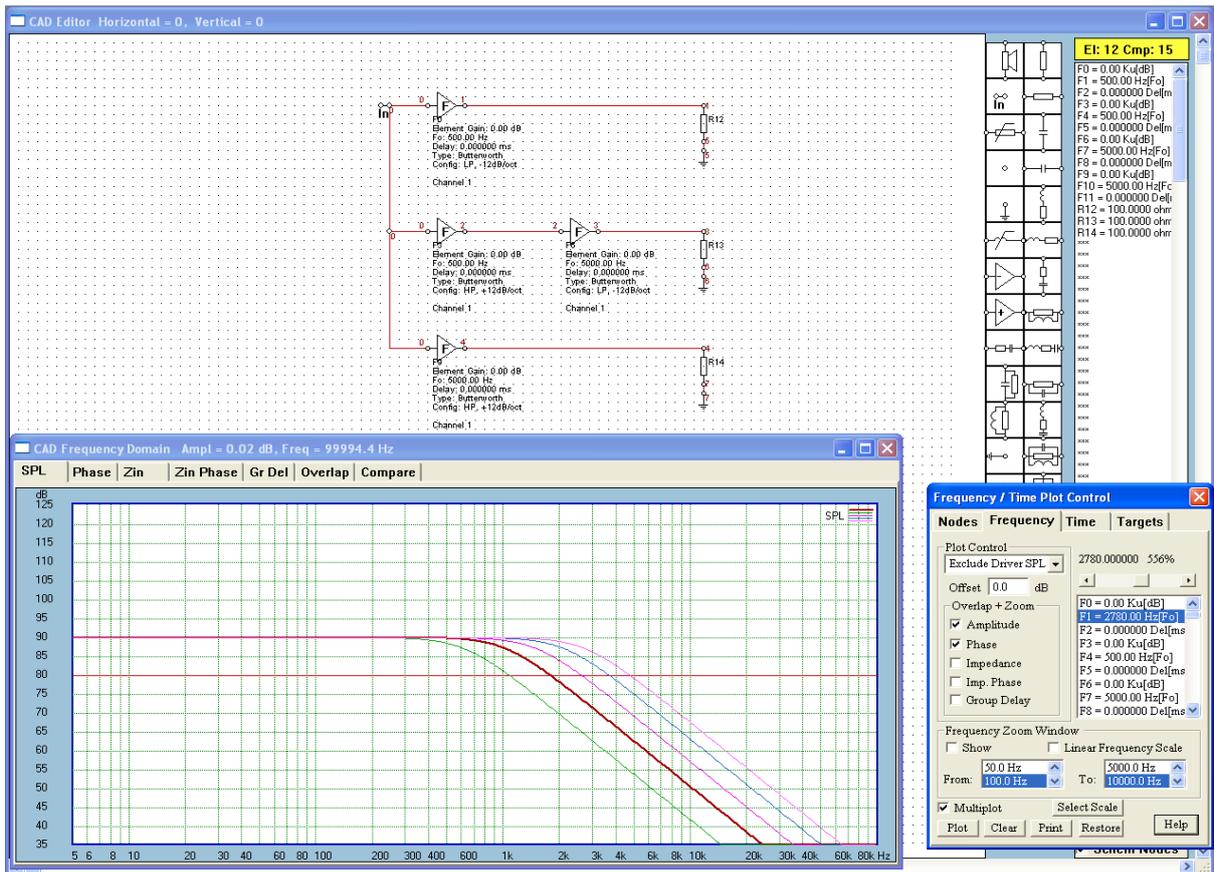
Fig 9.56. Adjusting cut-off frequency with a slider.

9.44

Here is how it would work:

1. Built your 3-way crossover using F-components.
2. You can now **adjust cut-off frequencies, gains and Q-factors with the slider**, after highlighting this parameter in the component list box – see Fig 9.56
3. You can also perform initial optimization of the MC crossover.
4. Once you are happy with the result, you can easily built your final crossover with discrete components by selecting corresponding filter parameters when populating the schematic with built-in filters.

You MC can also be optimized. The F-component parameters can optimized for gain, cut-off frequency and Q-factor. Finally, the F-component can be used with Digital Filter and Digital Equalizer functions.

Using The DCX2496 Digital Crossover.

The MC concept within the program has been developed to enable an alternative transition path from the traditional discrete/passive components of the crossover to digital-type crossovers. You know already about SoundEasy Digital Filter and the powerful Digital Equalizer functions, utilizing your PC's soundcard. The DCX2496 option, will enable you to switch off your PC and use only the DCX as the crossover. The digital crossover technology is currently a lot more expensive than simple passive crossover. However, it offers some features, that the traditional passive crossovers are lacking. One of those features is ability to add a broadband delay to any channel.

Currently available DCX2496 device can be used with MC concept, but it is unable to function as the SoundEasy Digital Filter. An external digital crossover would have to be able to accept the concept of an acoustic channel defined by it's impulse response, IR, rather than the more traditional, mathematically defined filter parameters for a DSP engine. The situation is even more complicated for the Digital Equalizer function, because the IR of the Digital Equalizer is defined by 64000 coefficients (in SE V9.00). The other problem is with Linear Phase filters, which also can not be emulated by the DCX2496.

Typical usage of the MC concept and components would be to create a crossover with EQ and shelving circuits as you see fit, then proceed to optimize the modules and parameters, and finally audition your design using either your PC's soundcard or the DCX2496. This process is similar to the design and audition of a standard, passive, discrete component crossover. To audition the crossover, you can simply use your multi-channel sound card, or, if you have the Behringer DCX2496 Digital Crossover, you can use it for the same purpose.

Almost all F-components (or settings) of Modular Crossover can be transferred to Behringer Digital Crossover box – DCX2496. Please note, that MC has more options than currently available DCX2496 (with software version 1.14). Therefore some of the Bessel filter settings can not be transferred to the DCX2496 crossover. Additionally, the DCX2496 uses fixed frequency grid of 320 points between 20Hz and 20kHz. This may place some limitations on the accuracy of the cut-off frequency settings of your selected crossover frequencies and EQ centre frequencies. Please also note, that the finest Delay resolution setting on DCX2496 is 2mm, and the gain can be changed within +/-15dB.

You can transfer the settings manually, by simply keying them into the DCX2496 or electronically, by selecting menu option "Sych DCX2496 to MC" under the "Crossover Tools" main menu.

Before you can remotely flash the DCX2496 please make sure, your DCX2496 is connected to COM1 serial port and powered ON. Failure to do so, may hang up your PC with the program waiting for a response from a disconnected or inactive DCX2496.

The DCX2496 has a rich setting array. However, please bear in mind, that this device is originally intended as a PA/stage audio processor, with functionality more suitable for PA work than home Hi-Fi system. We will be adapting a sub-set of DCX2496 functionality, keeping in mind limitations described above. Here is a typical configuration of the DCX2496, that you will be working with.

1. Channel Configuration is set to STEREO, 3WAY system: LMH LMH (CH1,2,3 CH4,5,6)
2. Channels are linked. Therefore, changes applied to channels 1,2,3 will automatically be transferred to channels 4,5,6.
3. Input section of DCX2496 has flat frequency response and gain set to 0dB.
4. All frequency response processing functions are implemented in the output section of DCX2496.
5. Each DCX2496 output section has 1 HP filter and 1 LP filter.
6. Each DCX2496 output section has up to 9 EQ circuits.
7. Cut-off frequencies of DCX2496 crossover filters are set to "FREE" mode (not linked) to allow you to set them individually.
8. During flashing process, all outputs are muted first, then new settings are applied, and finally all outputs are unmated.
9. Channels with no filters selected in, will have flat frequency response.

The freedom of CAD pick-and-place method allows you to built your MC schematic in any order/way you like it. On the other hand, the DCX2496 allows you to built each of your processing channels by selecting ON/OFF fixed channel components. Therefore, some mapping process must take place, in order to associate MC schematic components with the available DCX2496 audio processing components.

When you double-click on the MC component, the program pops-up a dialogue box with all available settings for this component. At the bottom of the dialogue box you will find a short list box designed for nominating the DCX2496 channel, that this component belongs to. There are only three settings needed to choose from:
1. Channel 1
2. Channel 2
3. Channel 3

Here, you must nominate the channel, that this component belongs to. This way, the program can correctly transfer the settings AND configuration of your crossover to the DCX2496. Currently, you can use DCX2496 to emulate STEREO 2-way and STEREO 3-way crossovers.
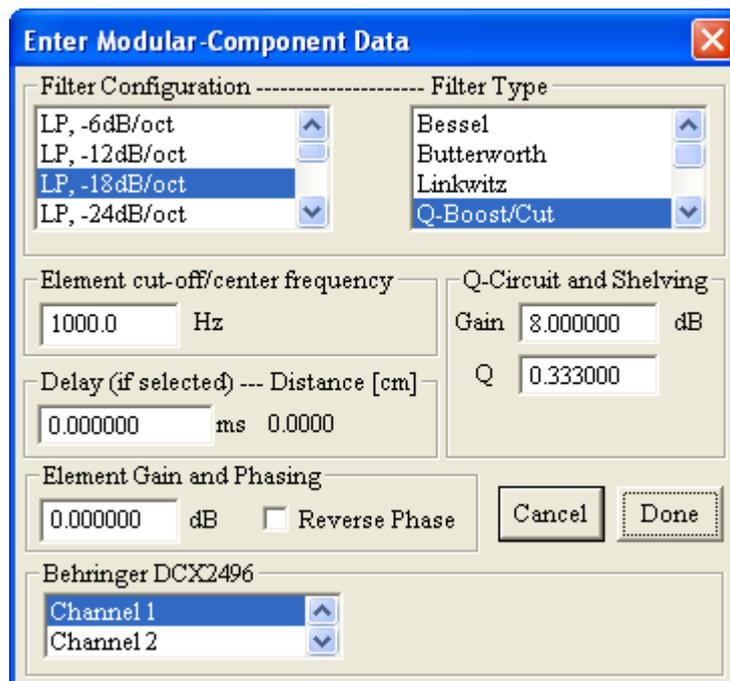
Figure 9.57. Behringer DCX2496 Channel assignment.

For instance, for a simple 3-way MC , you would allocate LP filter to Channel 1 and connect your woofer to it, BP filter to Channel 2 and HP filter to channel 3. Please remember, that DCX2496 does not implement Band Pass filters. You must construct such filter from a High-Pass + Low-Pass combination of the F-components.

Please make sure, that you are comfortable with the process of allocating the F-components to a specific DCX2496 channel. The process is very simple, but very important for setting the DCX2496 correctly. For instance, if you make an error and allocate a HP filter to a channel other than your tweeter, you will leave the tweeter totally unprotected from low frequencies – the tweeter will receive full audio spectrum via channel with FLAT low-end frequency response.

When you select menu option "DCX2496 Default" under "Crossover Tools". The DCX device will be set to the following values:

1. Channel Configuration is set to STEREO, 3WAY system: LMH LMH (CH1,2,3   CH4,5,6)
2. Channels are linked. Therefore, changes applied to channels 1,2,3 will automatically be transferred to channels 4,5,6.
3. Input section of the DCX2496 has flat frequency response and gain set to 0.0dB.
4. All output sections of the DCX2496 have gains set to o.0dB.
5. All frequency response processing functions are implemented in the output section of the DCX2496.
6. Output 1,4 has HP = 30Hz and LP = 300Hz filters.
7. Output 2,5 has HP = 300Hz and LP = 3000Hz filters.
8. Output 3,6 has HP = 3000Hz and LP = 20000Hz filters
9. All EQs, Dynamic EQs and Delays are OFF.
10. Polarity of all outputs is NORMAL.
11. Cut-off frequencies of the DCX2496 crossover filters are set to "LINKED" mode.
12. During flashing process, all outputs are muted first, then new settings are applied, and finally all outputs are unmated.

Using the "DCX2496 Default" menu option is a quick way of testing the RS232 link between your PC and the DCX device. When you select the "DCX2496 Default" menu option, you should see all red "Mute" LEDs on the DCX2496 front panel lighting up for a short time, and then going off, so the correctly filtered signals would appear on the outputs of the DCX2496 device. You can easily perform this checks with SoundEasy. The RS232 link is only active when the actual data is transferred from the PC to the DCX2496. When this process is completed, the DCX2496 becomes virtually a "stand alone" device. You can then use Spectrum Analyser screen to perform the above tests with the built-in White Noise signal generator. An alternative method for testing the DCX2496 would be of course the Digital or Analogue measurement screens. An example of DCX2496 SPL measured with SoundEasy MLS screen is shown below. Sampling frequency for channel H was set to 96kHz.
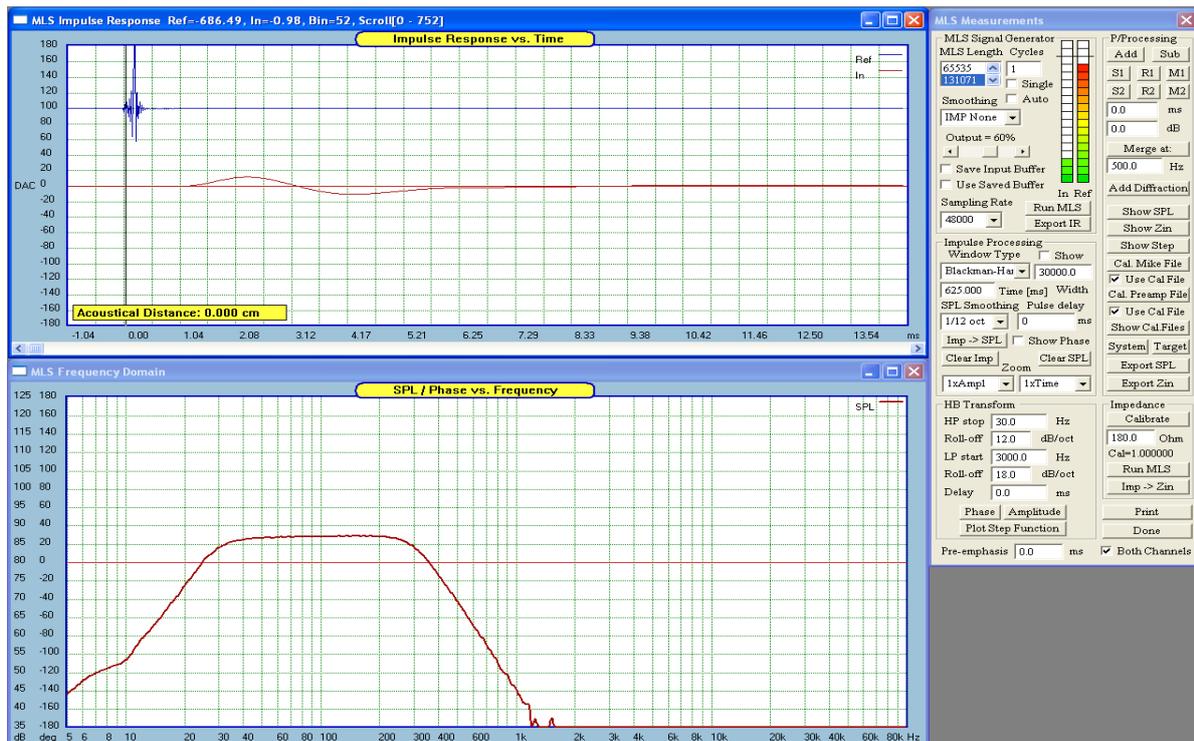

Fig 9.58. DCX2496 Default setting for L channel (IR + SPL).

9.47
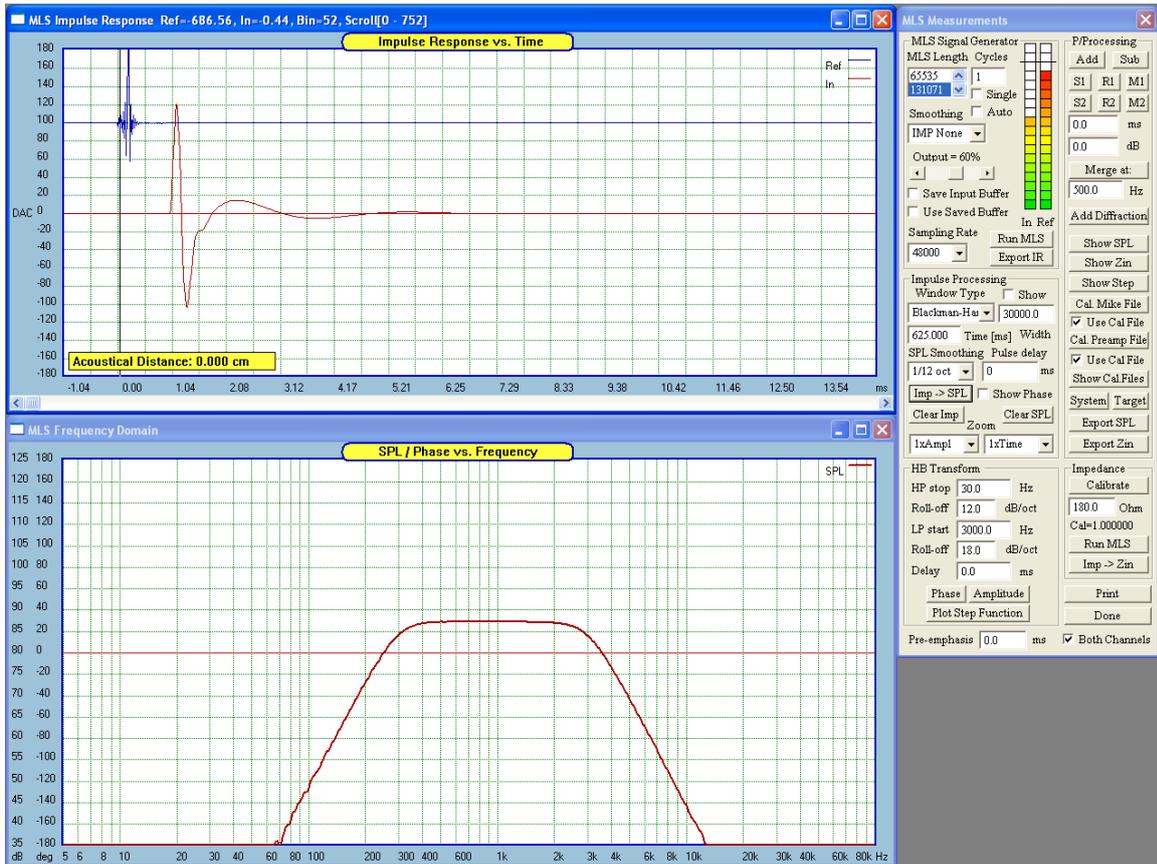
Fig 9.59. DCX2496 Default setting for M channel (IR + SPL).
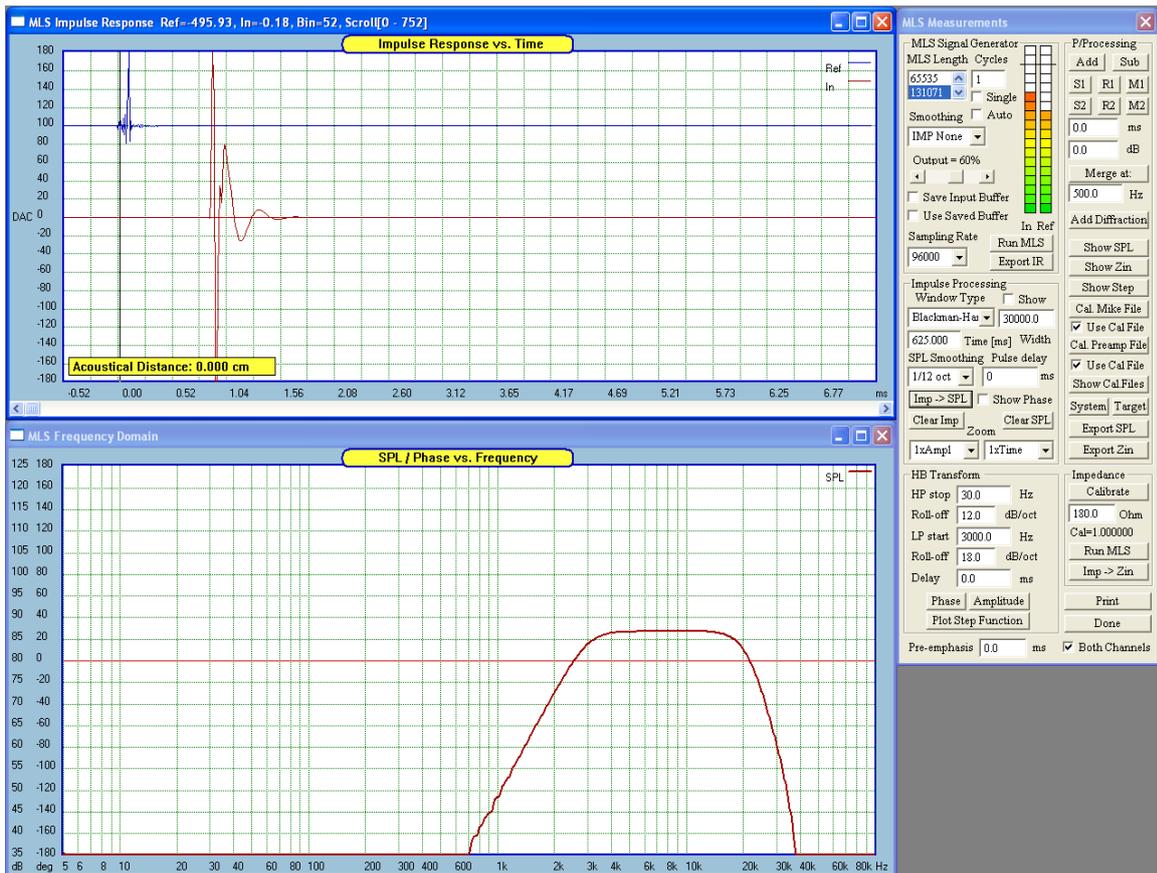
Fig 9. 60. DCX2496 Default setting for H channel (IR + SPL).
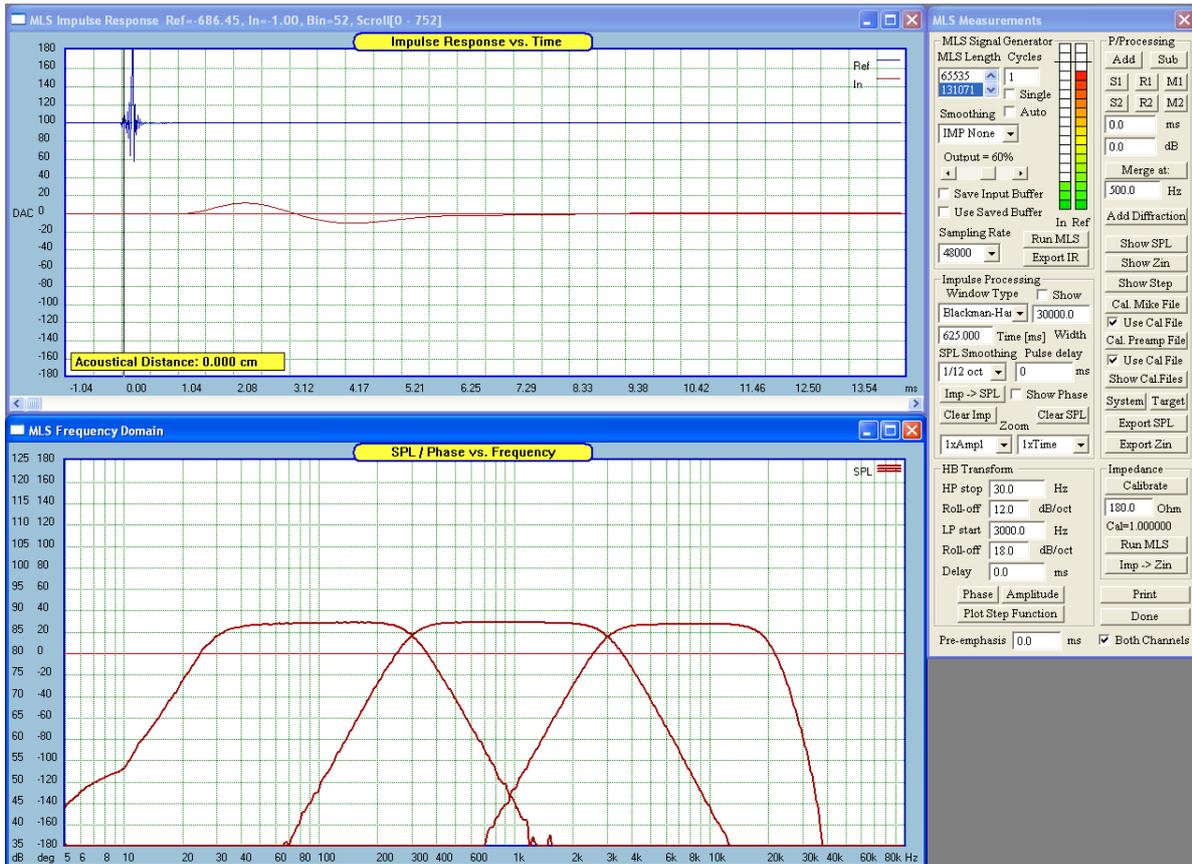
Fig 9. 61. DCX2496 Default setting for L+M+H channel (SPL).

Please note, that SoundEasy transfers MC "Gain" as the combined DCX2496 output section gain – see DCX block diagram.

Disclaimer:

SoundEasy V13.00 is compatible with Behringer DCX2496, software version 1.14.

Please note, that, depending on the available technical information, Bodzio Software Pty. Ltd. makes every effort to maintain compatibility with DCX2496 software versions. However, we can not guarantee, that all Modular Crossover settings, will be successfully transferred to DCX2496. This is due to variations in DXC2496 software implementations.

Bi-Quad Filters

$$H(s) = K \, (s + z1)(S + z2) / [(s + p1)(s + p2)]$$

This equation for transfer function H(s), is called a biquadratic equation, or a biquad for short. The **zn** terms in the numerator represent zeros and the **pn** terms in the denominator represent poles. Sallen-Key filters, state-variable variable filters, multiple feedback filters and many other types are all biquads. There also is a "biquad" topology, and three of such filters have been implemented in this release of the program.

Bi-Quad filters are selectable from the built-in "Filter Selector" dialogue box, using **"Filter Type"** list box. There are three types of Bi-Quads available. Implementation is based on OPAMPS and as the filter includes capacitors and resistors, you will need to nominate Ro, Co, Qo (Q-factor), Go (gain at resonance) and high-pass, low-pass frequencies. Please note, that low-pass F3dB is used as a band-pass filter centre frequency.

For a Bi-Quad, as Fc changes, the bandwidth stays constant, but the Q value changes. Thus, if you change Fc in the frequency domain, as Fc increases, the Q value increases and as Fc decreases, the Q value also decreases. It allows very high Q values, it can be configured in a 3 or 4 amplifier configuration, and it too is less sensitive to external component variations.

9.49
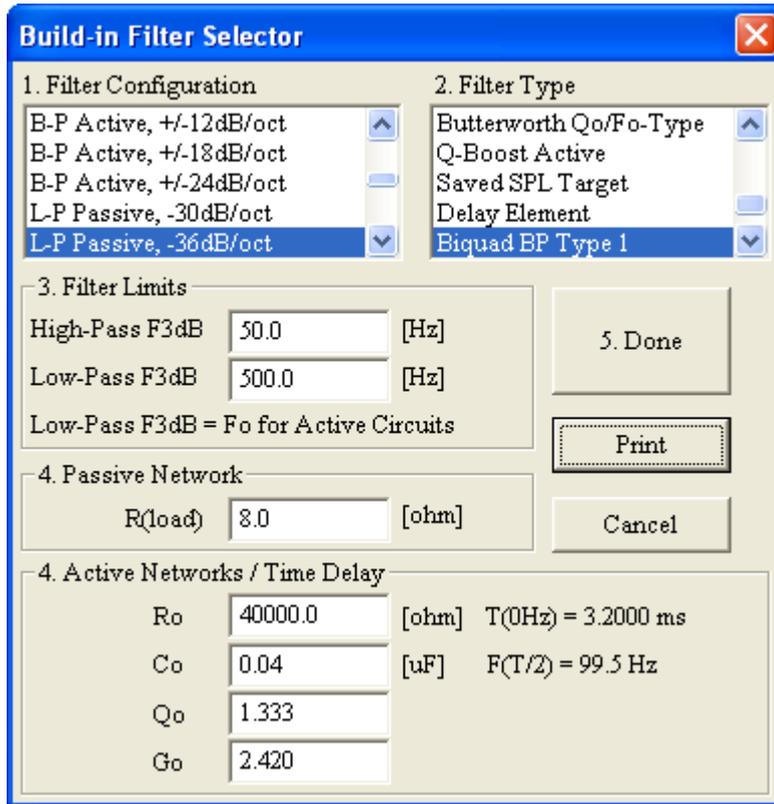
Fig. 9.62. Nominate Ro, Co, Qo (Q-factor), Go (gain at resonance)

Bi-Quad Type 1

Bi-Quad Type 1 is a band-pass filter. Selectable parameters are center frequency - Fo, quality factor - Qo and gain - Go.
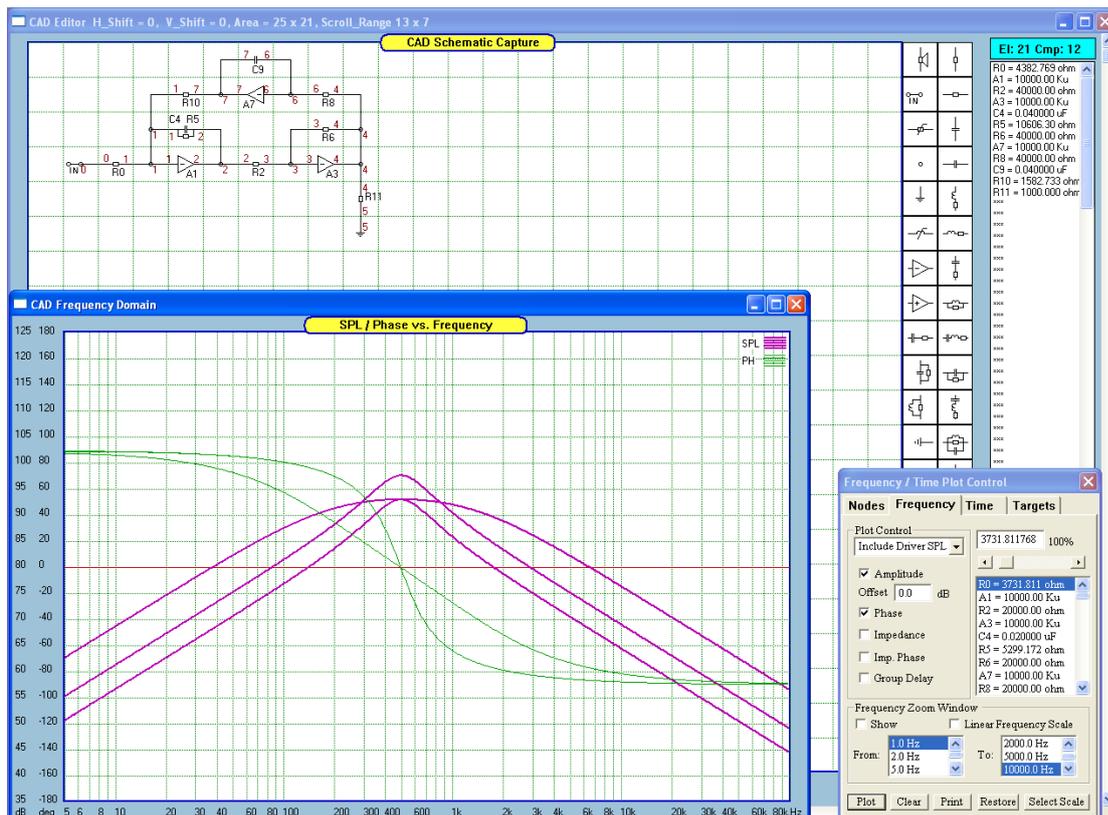

Figure 9.63 Examples of transfer function of Bi-Quad Type 1.

9.50

Bi-Quad Type 2
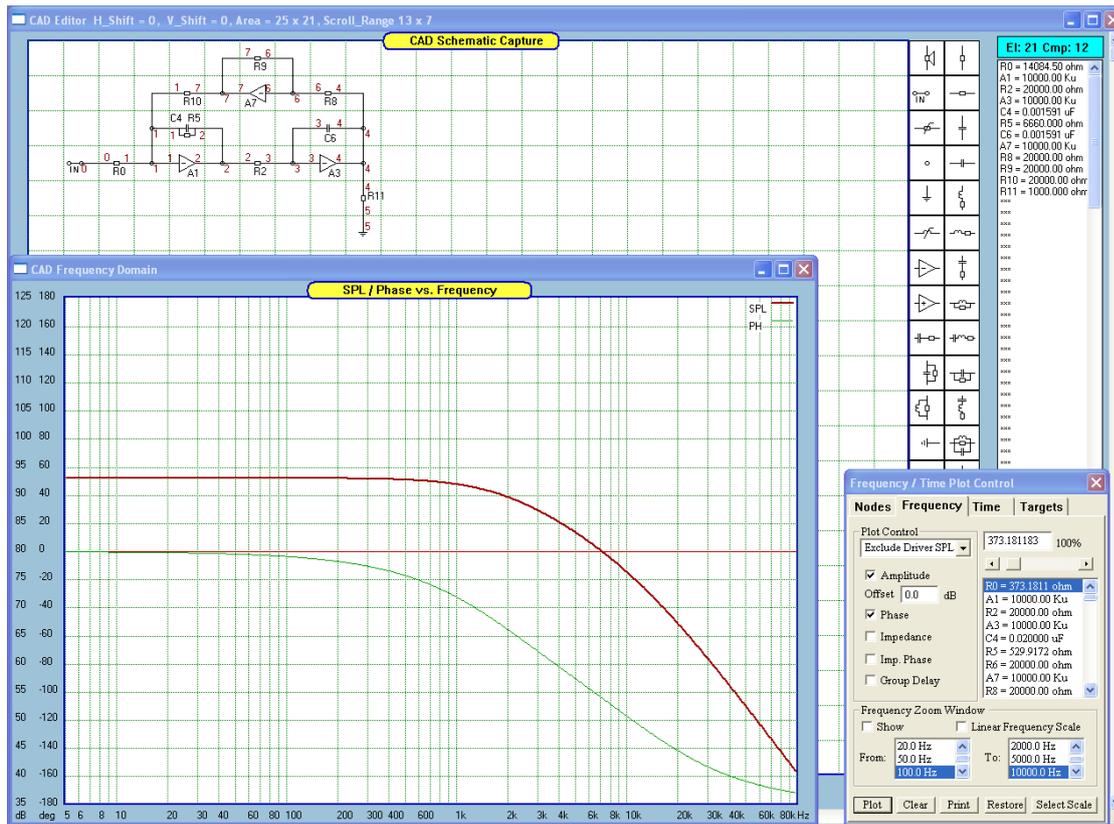
Figure 9.64 Examples of transfer function of Bi-Quad Type 2.

Bi-Quad Type 3
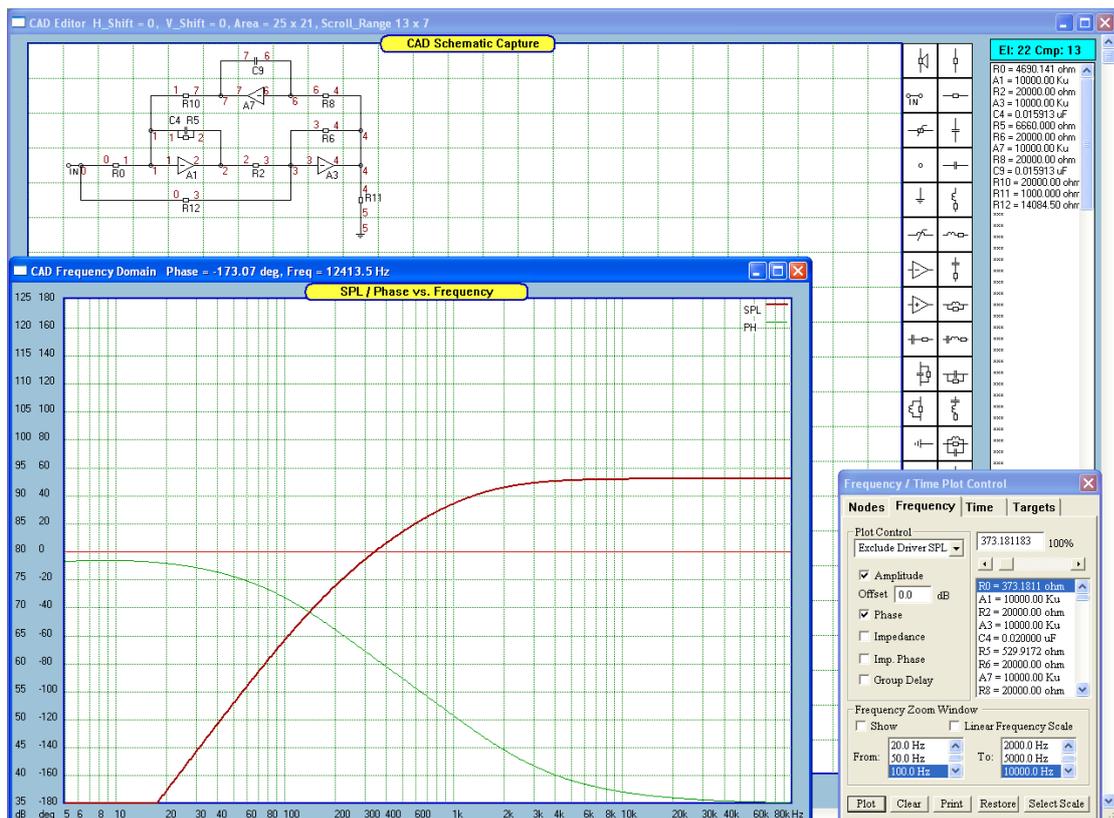
Figure 9.65 Examples of transfer function of Bi-Quad Type 3.

9.51

Linkwitz, Butterworth Asymmetrical Filters

At some occasions, you may need to use asymmetrical filters directly or, as templates/targets for your circuit to be optimized to. Asymmetrical filters can be created very easily in the program, but two such configurations are built-in for your convenience. These are: +12/-24dB/oct and −24/+12dB/oct active filters and targets.
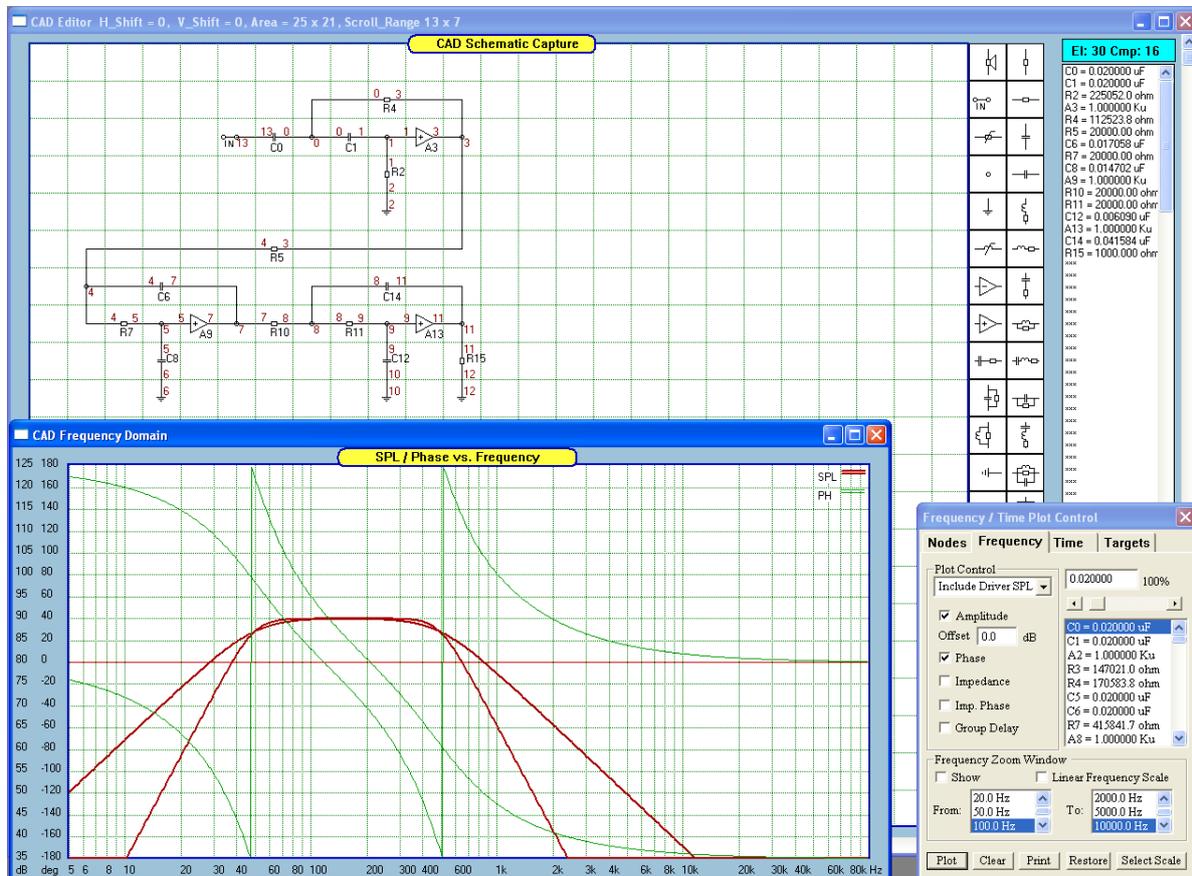
Figure 9.66. Linkwitz, Butterworth Asymmetrical Filters.

Linkwitz-Riley "Transformer"

When the subwoofer system is designed with the size in mind, it may happen, that the required enclosure volume causes a pronounced peak in response. This is an obvious sign, that the box is too small for the selected driver (its Vas). However, the peak can still be equalized and the low-end boost provided at the same time. The circuit recommended for this job is the Linkwitz-Riley filter and its example is shown on Fig 8.25. This circuit provides bass boost and correction for the peak. Design formulas are as follows:

1. Choose f_0, Q_0, f_p, Q_p ,

2. $k = \dfrac{\dfrac{f_0}{f_p} - \dfrac{Q_0}{Q_p}}{\dfrac{Q_0}{Q_p} - \dfrac{f_p}{f_0}}$,

3. Choose C_2 ,

4. $R_1 = \dfrac{1}{2 * \pi * f * C_2 * \left[2 * Q_0 (1 + k)\right]}$,

5. $R_2 = 2 * k * R_1$,

6. $C_1 = C_2 * \left[2 * Q_0 (1 + k)\right]^2$,

7. $C_3 = C1 * \left(\dfrac{f_p}{f_0}\right)^2$,

8. $R_3 = R_1 * \left(\dfrac{f_0}{f_p}\right)^2$

9.52

Example presented on Fig 8.24 below, shows the Linkwitz-Riley filter calculated for Qo=2, fo=70, Qp=2, fp=25 and C1=10nF. Press "Load Filter" to have the circuit displayed.
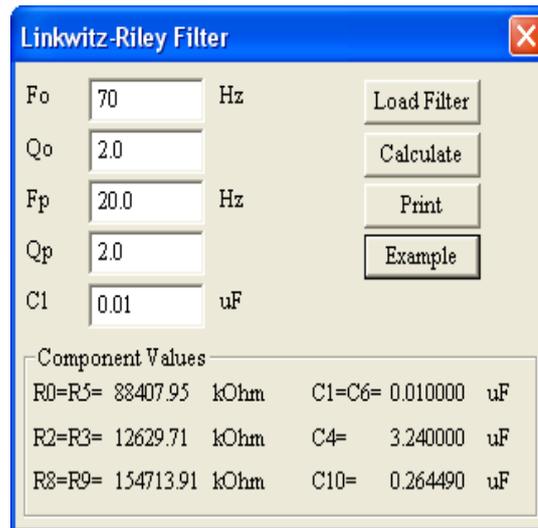
Figure 9.67. Linkwitz-Riley 'Transformer".
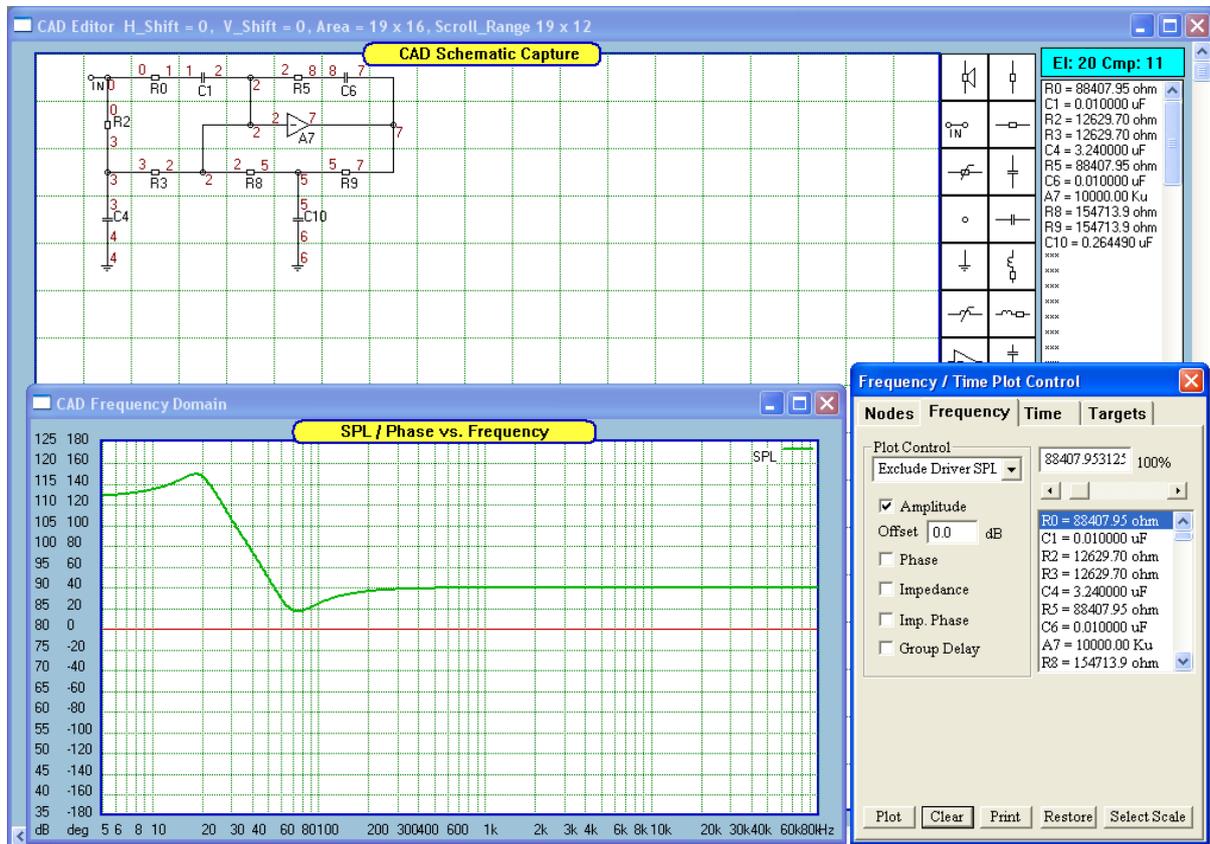
Figure 9.68. Frequency response of the L-R Transformer

Time Delay Circuit

As simple, first-order time delay circuit, is also available from the built-in networks. The delay is controlled by the Ro and Co components selected from "Filter Selector" dialogue box. The circuits can be easily cascaded, to obtain larger delays. The performance of the time delay circuit shown on Figure 8.27 can be estimated from the Filter Selector dialogue box. Entering values for Ro and Co will modify the time delay at To=0.0sec and also the frequency, F(T/2), at which the time delay falls to half of To value.

Both values: To and F(T/2) are displayed on the Filter Selector dialogue box. This way you can estimate how many sections on the active delay circuit you will need to obtain the required delay at the desired frequency.
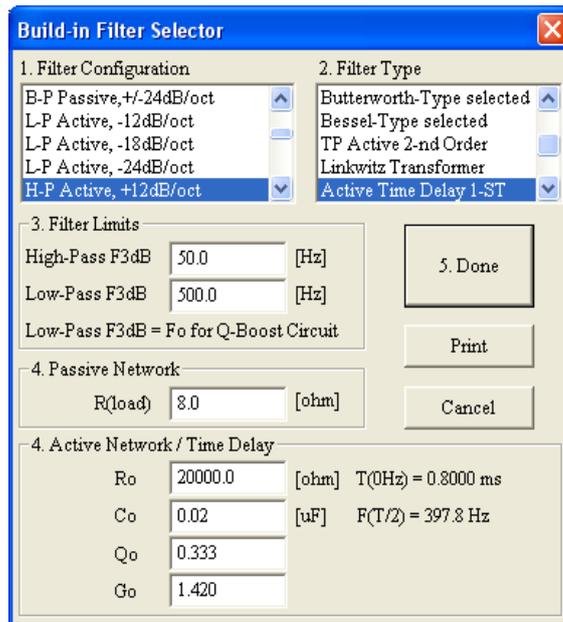
Figure 9.69. Time Delay circuit

**In the example on Figure 9.70, the goal was to obtain 0.8ms delay at 100Hz.**
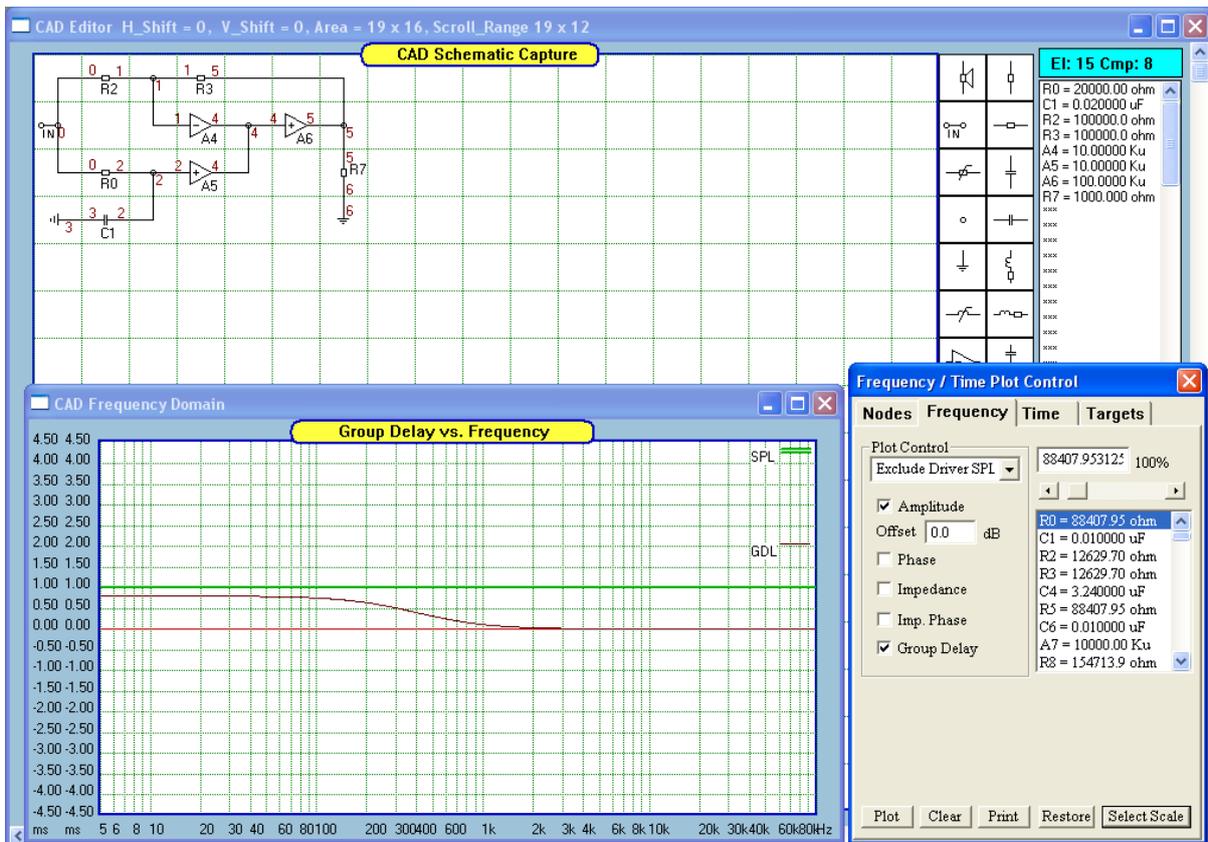
Figure 9.70. Built-in time delay circuit.

Notched Crossover / Filter

Notched filters are characterised by a null in the frequency response close to the transition frequency. Consequently, the SPL roll-off quite rapidly around the transition frequency. Additionally, the sum of the LP+HP filters sum to the flat response.

"k" is the ratio of lower notch frequency in the high-pass response to the crossover cut-off frequency, and is equal to the ratio of crossover cut-off frequency to lower notch frequency in the low-pass response.
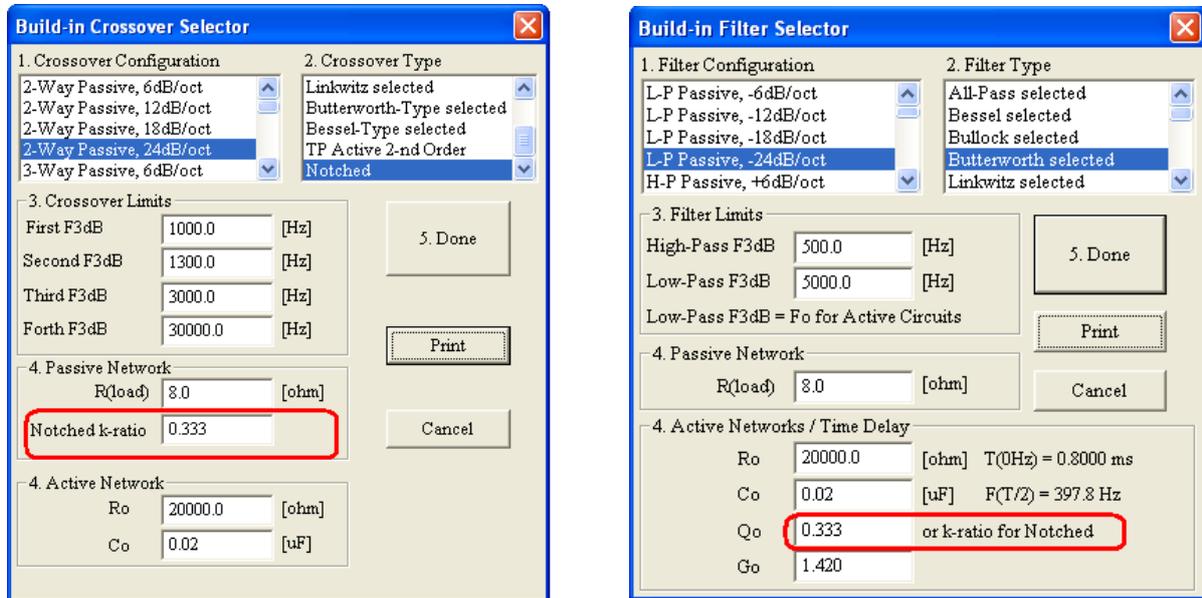
Figure 9.71. Notched k-ratio in the dialogue box.
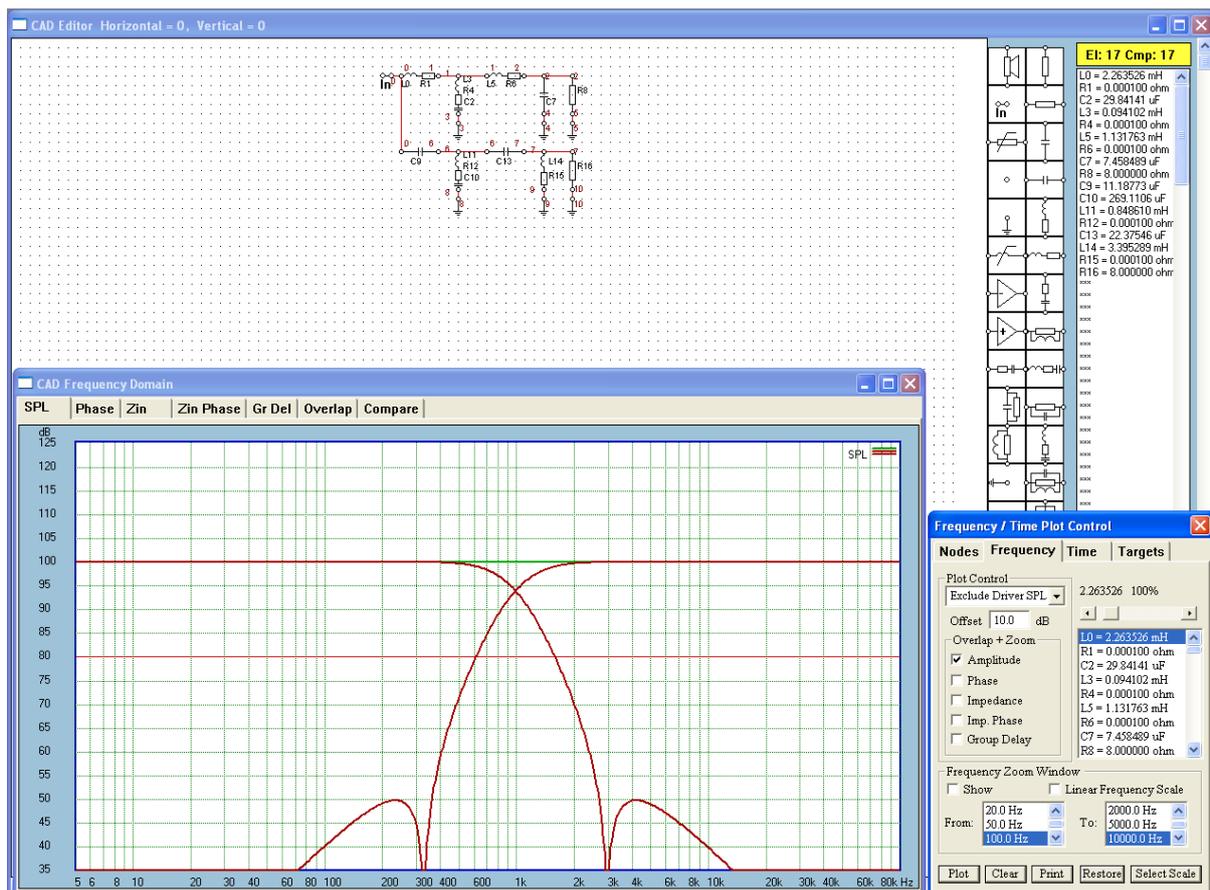
Figure 9.72 Example of a notched crossover.

9.55

Partitioned Convolution

Partitioned Convolution is understood to be a process of convolving a long Impulse Response with a series of short data blocks, so that a short latency between In-Out data processing can be obtained. Partitioned convolution is FFT-based, has fixed chunks and basically looks like this:

1. Obtain Impulse Response of the network.
2. Split the IR into blocks. Zero-pad them and FFT them. Store the FFT'ed blocks for later use in convolution process.
3. Then for every block of audio:
4. Zero-pad and FFT the input block
5. Store the FFT'ed block in a ring buffer (which has space for as many blocks as the IR has, too)
6. Multiply the FFT'ed input block from the input ring buffer with the corresponding FFT'ed response block and add all these.
7. iFFT the result from step 3.3
8. Save overlap. Add previous overlap.
9. Output block from 8.

When deciding on implementation of the Partitioned Convolutions, one must at one time make a choice between real FFT and complex FFT being used in the algorithm. Excellent comparison of the two, is presented in *"The Scientist and Engineer's Guide to Digital Signal Processing"* DSP book by Steven W. Smith (highly recommended reading). Implementation chosen for this program is based on complex FFTs. Here are major characteristics of complex FFT:

Time domain:
x[n] is complex, discrete and periodic, n runs over one period, from 0 to N-1

Frequency domain:
X[k] is complex, discrete and periodic, k runs over one period, from 0 to N-1
k = 0 to N/2 are positive frequencies
k = N/2 to N-1 are negative frequencies (blue colour on the chart below)
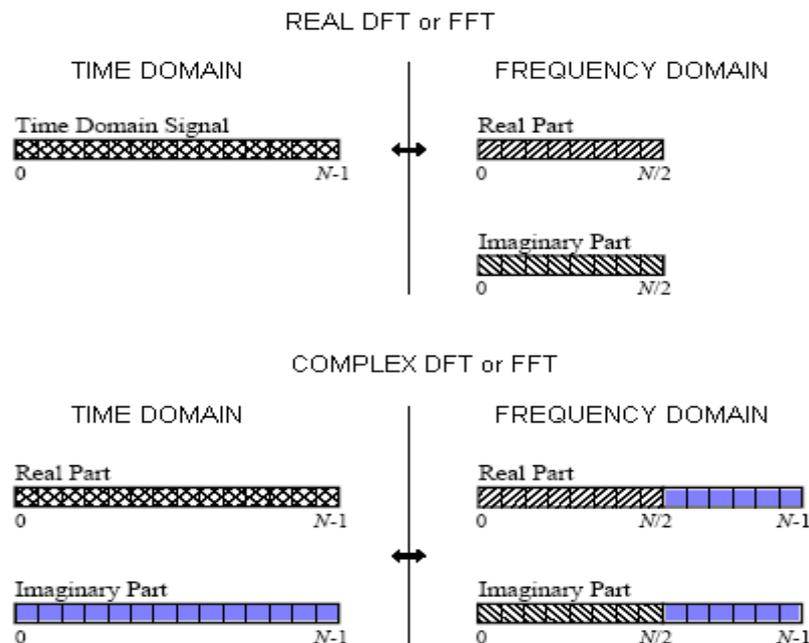
Figure 9.54. Complex/Real FFT comparison

In the complex Fourier transform, both arrays: $X[k]$ (frequency domain) and $x[n]$ (time domain) are of complex numbers. Given the above, the input data stream from the sound card, is treated as a series of real numbers, with the imaginary part set to zeros for all numbers.

Going from frequency to time domain, the cosine wave corresponds to *two* spectral values, a positive and a negative frequency. Both these frequencies have the amplitude value of ½. In other words, a positive frequency with an amplitude of ½, combines with a negative frequency with an amplitude of ½, producing a cosine wave with an amplitude of "1". Each value in the real part of the frequency domain contributes a real cosine wave *and* an imaginary sine wave to the time domain. At the same time, each value in the imaginary part of the frequency domain contributes a real sine wave *and* an imaginary cosine wave. The time domain is found by using all these real and imaginary sinusoids.

Generating Filter's Impulse Response.

This is not a trivial task for partition convolution process. In our typical application, we would design a filtering network using CAD system + associated functions, and as the outcome of this process, we would obtain a frequency response of the network. This is our X[k] array in the frequency domain. We assume, that our time-domain array will contain the IR, length of N, and we split the IR into four (4) segments. Given the X[k] (k=0…..N/2) as the starting point, we must:

1. Convert X[k] into the filter's Impulse Response array x[k] (iFFT process),
2. Then split the impulse response into the desired number of sections (or partitions),
3. Then convert the time-domain partitions into frequency-domain blocks (FFT process) suitable for performing frequency-domain convolution (multiplications in frequency domain).

Given, that we use complex iFFT, we may also provide the "negative frequencies" portion of the spectrum. So our X[k] must be calculated for positive and negative frequencies, assembled symmetrically around k=N/2, and presented to the iFFT algorithm as shown below. Another option is to calculate X[k] over the "positive frequency spectrum" frequency range only, and accept reduction of amplitude of each bin by half.
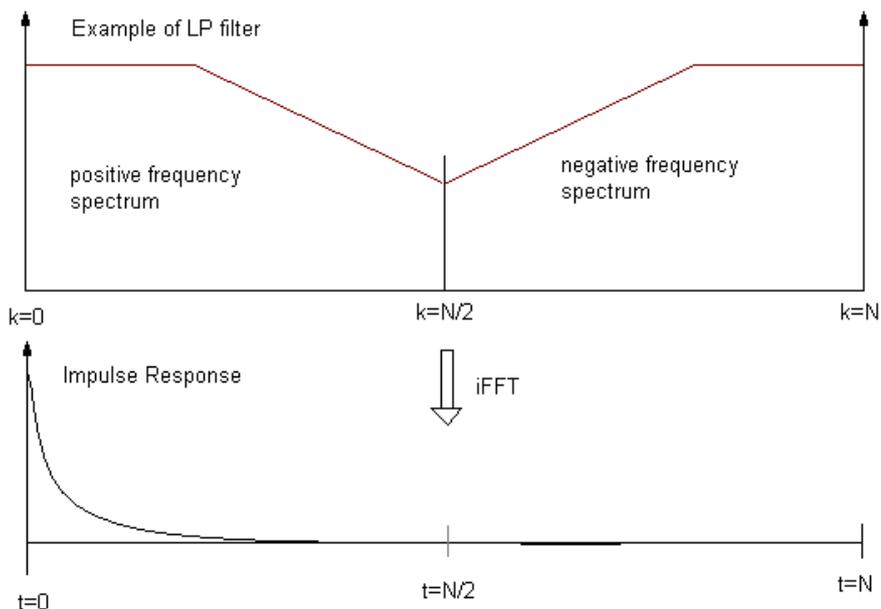

Figure 9.55 Generating IR from the spectrum

In the next step, the IR calculated above, is divided into 4 sections, each one has the length of N/4, as shown below.
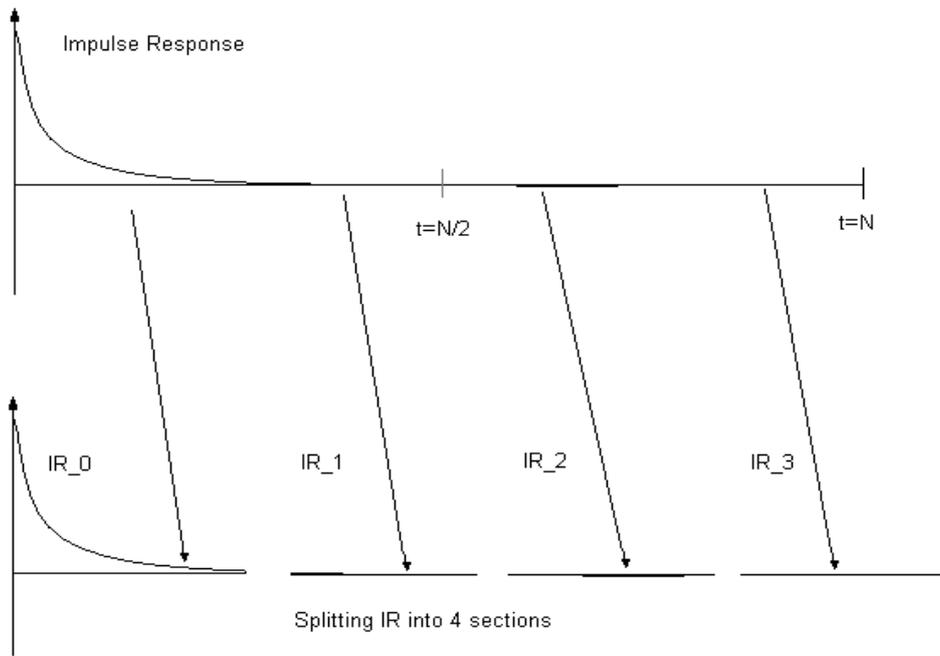
Figure 9.56. Splitting IR into four sections.

In the final step, each IR section (IR_0, IR_1, IR_2, IR_3) is zero-padded to N/2 and FFT'ed back into frequency domain.
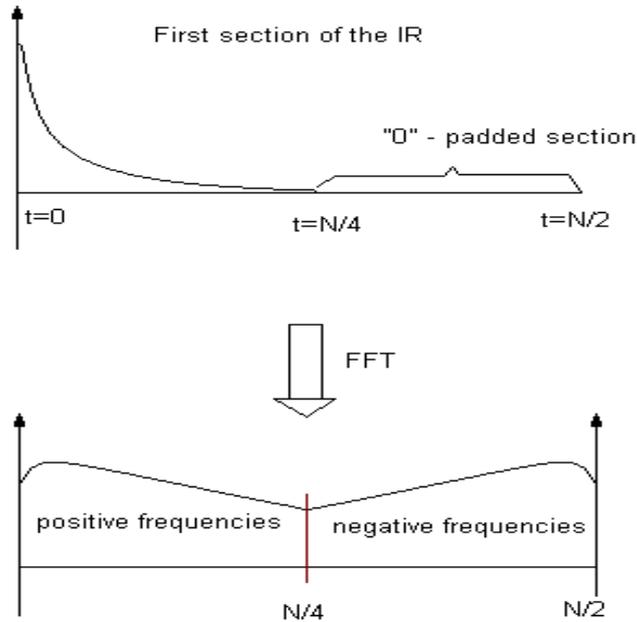

Figure 9.57 Generating frequency domain for each of the previously obtained sections of IR.

All four sections of the IR are treated the same way: they are zero-padded to N/2 and FFT'ed back into the frequency domain. Now, each IR section is ready to contribute to the actual convolution process, of being multiplied with pre-processed input data blocks.

Processing Input Data – Ring Buffer

While running, the algorithm reads-in input data blocks of N/4 length. Each block is zero-padded to N/2 and FFT'ed before it is placed into a ring buffer.
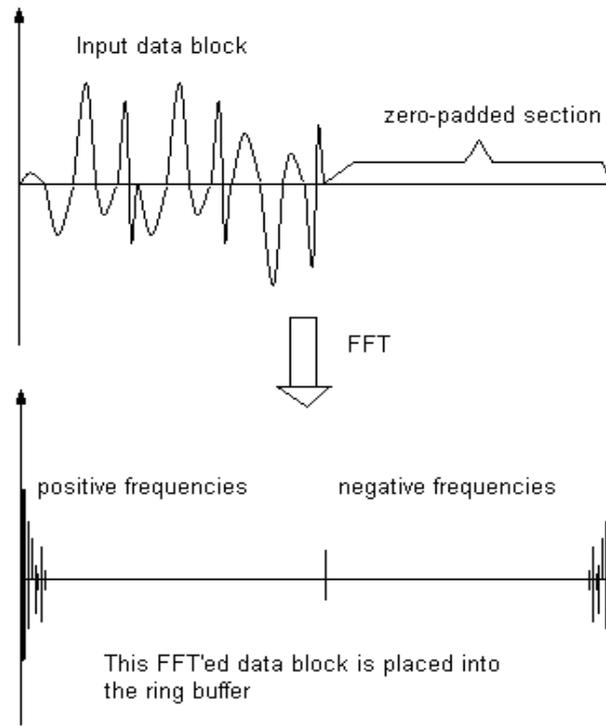

Figure 9.58. Obtaining spectrum of the input blocks.

The ring buffer has 4 sections: Section_0, Section_1, Section_2 and Section_3. The operation of shifting (or rotating) data in the buffer is explained in following steps:

Section_1 -> Section_0, old Section_0 is now disregarded.
Section_2 -> Section_1
Section_3 -> Section_2
New Data -> Section_3

It is easy to observe, that Section_3 has the latest data block, Section_2 has the data read-in during the immediate previous cycle, and so on….As the new time-domain data block is read-in and FFT-processed into frequency domain, all previously stored, frequency domain blocks are shifted back in time. Data from Section_0 is simply disregarded after it had been used in the last convolution cycle.

Actual Convolution Process

Convolution in frequency domain is just a multiplication of the FFT'ed data blocks from the ring buffer with the corresponding FFT'ed IR blocks.

IR_0 x Section_3
IR_1 x Section_2
IR_2 x Section_1
IR_3 x Section_0

The results of the above multiplications are added together and iFFT'ed back into the time domain. All data located between N/4 and N/2 is called "Overlap". It is saved for the next cycle of processing. Now, the "Overlap" from the previous processing cycle is added to the output data block, and the output block is now ready to be send to the output of the sound card.
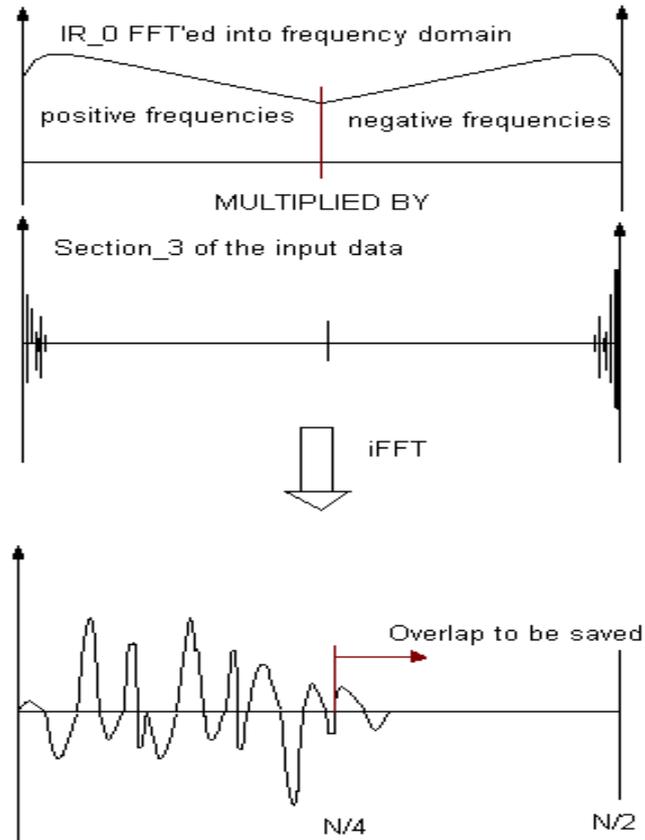
Figure 9.59. Dealing with overlaps.

The process described above, implements partitioned convolution with ONE FFT process for input data, and ONE iFFT process for output data. There are variations of this process available, where multiple iFFT's are used for assembling the output data.

The Impulse Response has been partitioned into four sections. However, nothing stops us from "chopping" the IR into 8, 16, 32, 64… and more sections. If you keep the IR length the same for each attempt, the effect of increasing the number of partitions will be reflected as ever shorter latency between input and output of the process. In practical terms, you will eventually hit the latency limitations imposed by the combination of your operating system and sound card drivers. On the other hand, the processing burden shifts from FFT to multiplications/additions during the convolution process.

If one prefers to keep the latency the same, one can use longer and longer Impulse Response. As you increase the number of partitions. For instance, with the input data blocks of 2048 bins and 64 partitions, you can use 2048 x 64 = 131072 bin long Impulse Response. Given the sampling frequency of 48kHz, this would give you 2.7 seconds long IR. This is something you may contemplate to use for de-convolving your room. But for filtering purposes, 16-32 IR partitions for the conditions outlined above should be sufficient.

Current implementation aims for a latency less than 150ms. Measured value of latency with Delta 410 sound card, input data blocks of 2048 samples, sampling frequency of 48kHz, and WinXP operating system is about 100ms. This is deemed acceptable for viewing video pictures with audio processing done on the PC using the above process.

Partitioned Convolution and Linear Phase Filters

Zero Phase – A filter is said to be "zero-phase" filter, if it's impulse response (IR) has left-right symmetry around sample number zero.

Linear Phase - A filter is said to be "linear-phase" filter, if it has left-right symmetry, but around some point other than zero. FIR filters are not automatically linear phase. Linear phase FIR filters require symmetry of the IR, with the "original" sample in the exact middle.

Nonlinear Phase - A filter is said to be "nonlinear-phase" filter, if it does not have left-right symmetry. We may also call this type of filter a "minimum-phase" filters.

The IR symmetry, required for linear-phase filtering has some implications for low-latency, partitioned convolution approach. This issue, can be explained using a 2-nd order, high-Q, low-pass filter, with it's frequency response shown below.


Figure 9.60 Example of a very high Q, low-pass filter frequency response.

If the above filter was to be implemented as a "minimum-phase" filtering device, it's impulse response would be asymmetrical, and it would start at sample "0" – as shown on the Figure 9.61 below – the rising edge of the blue curve. We would not be interested in what happens before sample "0", and this would imply, that the partitioned convolution process can start right at sample "0".

The top section of Figure 9.61 shows total impulse response, segmented into 8 blocks. It is a "zoomed-out" view of the complete IR. It is observable that, our IR starts right at the beginning of "Segment 0". The IR is rather short, so other segments do not show any data.

If we were to implement the above filter as a "linear-phase" device, the IR would have to be symmetrical around some selected sample. Selecting such sample is not a trivial task. Please remember, that we are trying to secure the ability of convolution process to accept: (1) long Impulse Responses, and (2) required minimized latency.

For the "minimum-phase" IRs, the above requirements are met automatically. This is because the start of IR is at the left-most sample. Consequently, this would provide us with the longest possible IR, and the shortest possible processing latency.

Please note the shape of Real part of the IR (blue curve) and the Imaginary part of IR (green curve) on Figure 9.61. The are both asymmetrical. This is typical for "minimum-phase" filters.

9.61
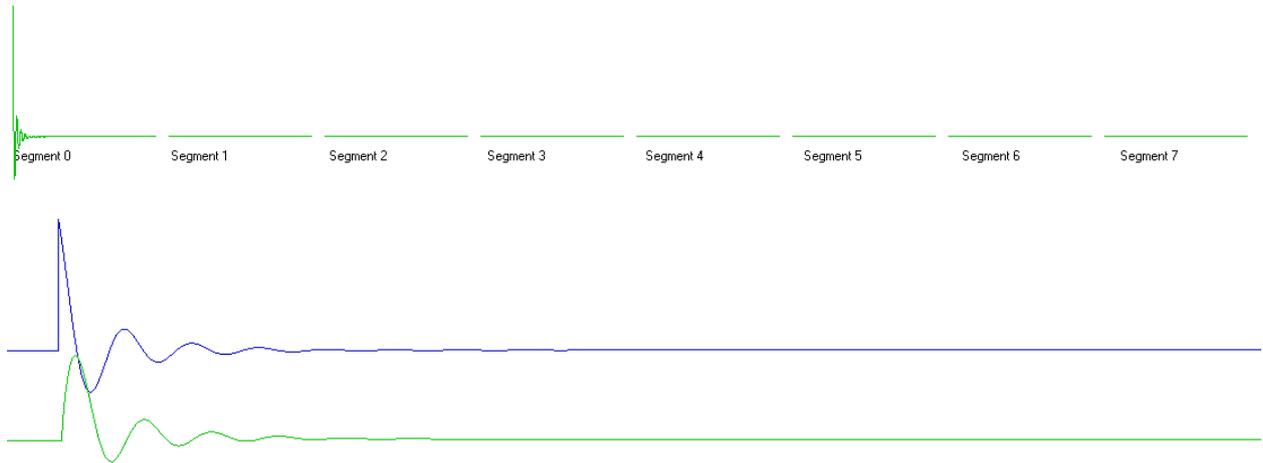
Figure 9.61 Impulse response of a "minimum-phase" or "non-linear phase" filter.

For the "linear-phase" filter implementation, we must remember, that it's IR must now be symmetrical, therefore, the longest IR for fully symmetrical condition will be met when the IR is placed exactly in the middle of the sample data space. As a side effect of this, the latency will be increased significantly.

Example below on Figure 9.62, shows the IR of the same low-pass filter, converted to "linear-phase" and shifted to the right by about 1000samples in Segment 0. Please note the shape of Real part of the IR (blue curve) and the Imaginary part of IR (green curve) on Figure 9.62. The Real part is fully symmetrical around sample 1000, and the Imaginary part is anti-symmetrical around sample 1000. This is very characteristic picture for "linear-phase" filters.
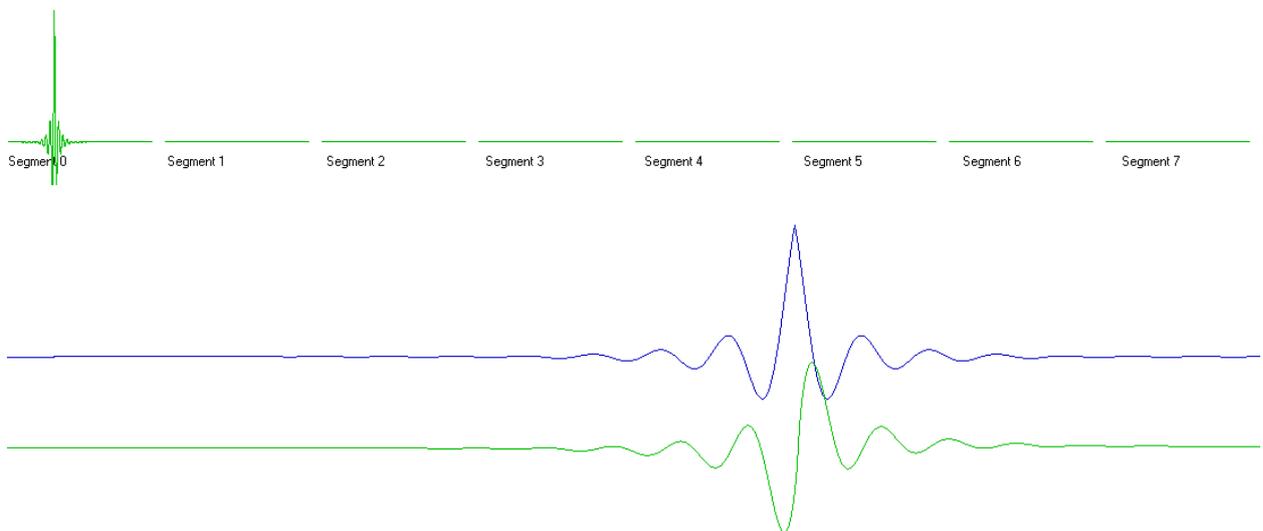

Figure 9.62 Impulse response of a "linear-phase" filter version.

If we work with the segment length of 2048 samples, and sampling frequency is 48kHz, then the sampling period is 20.83usec. Shifting the IR to the centre of an 8-segment long IR, will introduce 4 x 2048 x 20.83usec = 170msec of additional latency. The program has a built-in feature, that will shift the IR to the right by 60msec, to allow for symmetrical IRs. Additional delays can be set from "T-Element" time delay data field.

9.62

If you have a minimum-phase filter transfer function H(jw):

$$H(j\omega) = A(\omega) + jB(\omega)$$

$$D(\omega) = |H(j\omega)| = \sqrt{A^2(\omega) + B^2(\omega)}$$

Then you can implement the linear-phase transfer function G(jw) version of it by:

$$\text{Re}\{G(\omega)\} = D(\omega)\cos(\omega t)$$

$$\text{Im}\{G(\omega)\} = D(\omega)(-\sin(\omega t))$$

Both linear and partitioned convolution processes implemented in the program, use FIR filter approach. The FIR filters can usually be designed to be linear-phase (but they don't have to be.) A FIR filter is linear-phase if (and only if) its coefficients are symmetrical around the center coefficient, that is, the first coefficient is the same as the last; the second is the same as the next-to-last, etc. A linear-phase FIR filter having an odd number of coefficients will have a single coefficient in the center which has no mate.

A FIR filter will exhibit a linear phase if its IR, h[n] satisfies either one of the following two symmetry conditions:

$$h[n] = h[N-1-n], \qquad 0 \le n \le N-1 \quad \text{(symmetric)}$$

Or

$$h[n] = -h[N-1-n], \qquad 0 \le n \le N-1 \quad \text{anti-symmetric}$$

There are four possible categories of linear phase filters according to their symmetry and length:

Type 1 - symmetric, odd length
Type 2 - symmetric, even length
Type 3 - antisymmetric, odd length
Type 4 - antisymmetric, even length

Examples of the four types of impulse response sequences are shown in Figure 9.63.
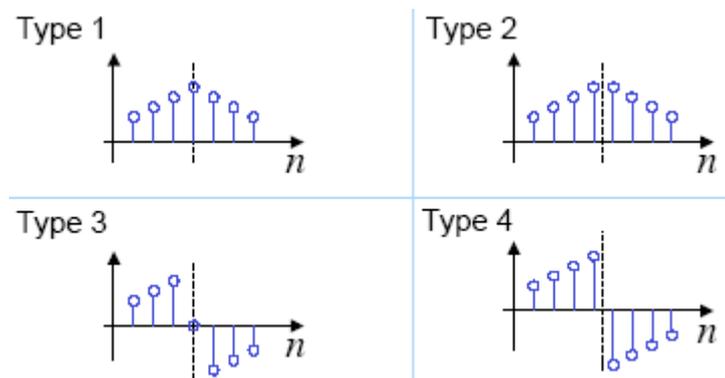
Figure 9.63. Four common FIR filter types

Please note additionally that a Type 3 filter must have h [(N-1)/2] = 0.

Each FIR filter type has specific frequency domain characteristics.

A Type 1 filter may be used to implement any desired bandpass frequency response. Therefore, this is the configuration used in this program.

The symmetry condition of a Type 2 filter implies that it may not be used to define a highpass filter. Otherwise it may be used instead of Type 1 in cases where an even-length filter is preferable.

In contrast, antisymmetric filters (Type 3 and 4) cannot be used to implement any sort of bandpass filter. They are the only choice, however, for designing FIR differentiators, which are an important class of filters applicable to edge detectors in video processing.

When choosing "linear-phase" vs. "non-linear phase" implementation, please be aware of the latency penalty you will experience when opting for the "linear-phase" implementation.

If the circuit you working with has "linear-phase" components (meaning, you will incur **60ms delay per component**) and "minimum-phase" components (meaning, you will NOT incur any delays) there is obviously a problem of grossly mismatched processing delays in different channels.

To resolve this problem, the program has a built-in detection function, that discovers the "linear-phase" element and inserts 60ms delay chunk for each channel. As a result, ALL channels should exhibit the same amount of delay. For instance, consider the network shown on Figure 9.64 below. You will notice, that midrange (Node6) channel has 2 x T-Elements, one set up as "linear-phase" HP filter and the other set up as "linear-phase" LP filter. The other two channels (woofer Node 1and tweeter Node 2), incorporate only one T-Element filter per channel, woofer being "minimum-phase" and tweeter being "linear-phase". The detection function will introduce 60ms delay in all channels.
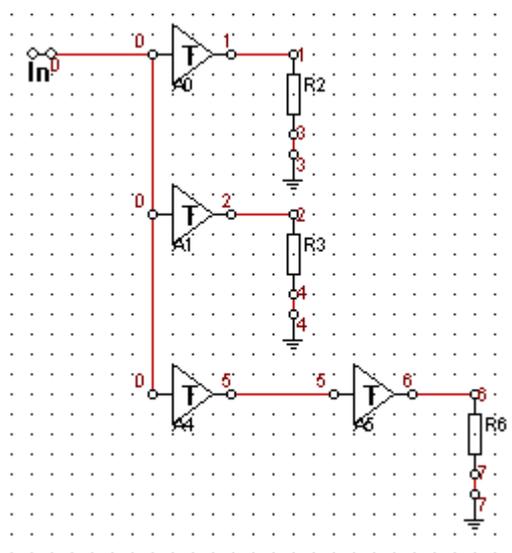

Figure 9.64. Linear-phase crossover will have the same delays in each channel.

In summary: the built-in detection function will :

1. Scan all filter channels for "linear-phase" filters.
2. If detected, it will add 60ms chunk of delays to each channel.
3. Please be aware of the following. You can manipulate the delay in each channel by entering your own delay (in ms) into T-Element "Delay" data field. This user-defined delay, will obviously affect the total delay of the channel. For instance, if you use the "linear-phase" component, and enter Delay = 5ms, the total delay for this component is 5ms + 60ms = 65ms.

9.64

General Latency Issues

In order to create a filter SPL curve of ANY desired shape, you will need an Impulse Response consisting of several thousands coefficients or more. This is the case of Digital Equalizer and Digital Room Equaliser. Calculating Convolution on such a long IR can be practically accomplished by using FFT/iFFT approach, also described as convolution in frequency domain. The rule here is simple – more complicated SPL curves, will require more coefficients to represent them. This approach is also known as "fast convolution", as one would be taking advantage of the FFT efficiency, increasing with it's length.

On the negative side, using long, efficient FFTs would imply, that incoming data stream is divided into similarly long blocks, thus creating a severe latency problem. Latency here is understood as the total input-output signal delay.

Here is what happens in more details. Assume, that sampling frequency is equal to 48kHz (or 20.83usec sampling time) and the input data buffer and FFTs are 2048bins long.
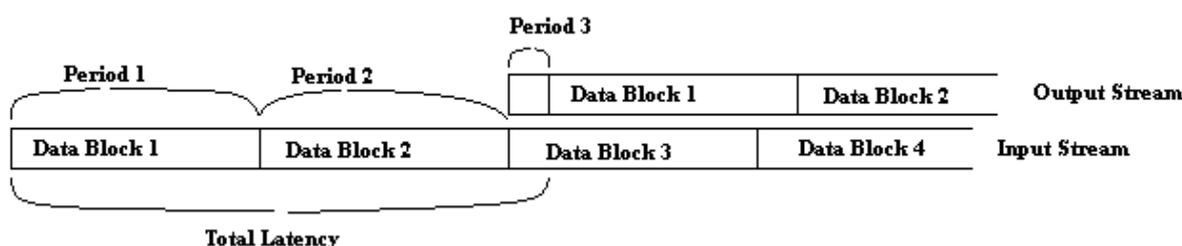

Figure 9.65 Explanation of processing latency.

Period 1 – Input data buffer is acquiring data stream. It will take 42.66ms to fill input data buffer. At the end of Period 1, the input data block 1 is ready for processing. Period 1 is typically 45% of the Total Latency.

Period 2 – During this period, input data is being convolved with the filter impulse response. Please note, that there is only 42.66ms available for all mathematical processing, otherwise, you lose the chance to output the calculated data right at the end of Period 2. Interestingly, since the input data block 1 is being processed, the sound card is free to acquire input data block 2 during Period 2, and keep repeating the process indefinitely. This approach is referred to as "double-buffering system". Period 2 is typically 45% of the Total Latency.

Period 3 – During this time, calculated data is actually transferred to the sound card's output buffer. At the end of Period 3, the sound will come out of the card's output. The length of this period is typically dependant on the size of the HW buffer in the sound card, and can be comparatively small – in order of 5-10ms. Period 3 is typically 10% of the Total Latency.

Total Latency = Period 1 + Period 2 + Period 3 = appr. 95ms

It is easily observable, that we are dealing here with somewhat conflicting requirements when attempting to minimize latency. On one side, you would select the input buffer size to be as short as possible. This is because 45% of the latency depends on the how long it will take to fill-in the input buffer. The other 45% is taken by the necessary calculation time.

On the other hand, if you shrink the acquisition time, you automatically shorten available time for mathematical processing (convolution). So, this is where the problem starts manifesting itself.

In addition, there is also another factor to consider. The 2048 coefficients may not be sufficient for accurately convolving your Impulse Response. What if your IR is 16384 bins long?. Well, in this case, you will need a Partitioned Convolution, however, the latency problem remains.

As you can see, the actual latency introduced by the sound card is rather insignificant, in comparison with the latency introduced by the complexity of the convolution in frequency domain process. Many WDM/MME sound cards introduce latency, that is very low in comparison to rest of the necessary audio processing.

Second-Order Yamanaka Crossover

Second-Order (12dB/Oct) Yamanaka crossover filters can be designed usin the built-in option selected as per Figure 9.66 below.

Crossover Configuration = 3-Way Passive 12dB/Oct
**Crossover Type = TP Passive 2nd, 4th Order**
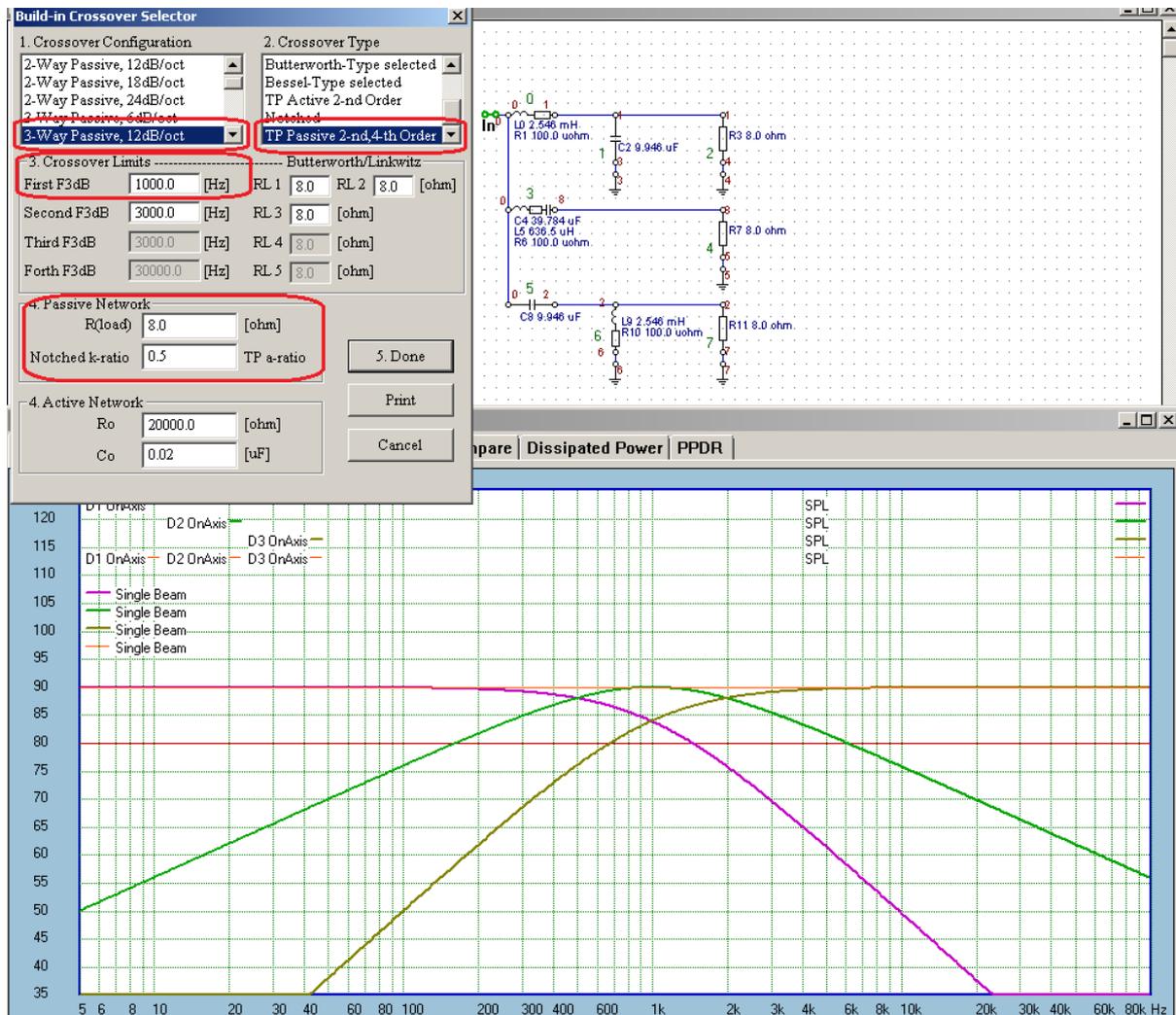
Figura 9.66 Yamanaka Crossover example.

Only the First 3dB frequency needs to be selected, and additional parameters are R(load), and TP- a-ratio. Figure 9.66 shows such filter with R(load) = 8ohms and a-ratio = 0.5.

Fourth-Order Delund Crossover

Fourth-Order (24dB/Oct) Delund crossover filters can be designed usin the built-in option selected as per Figure 9.66 below.

Crossover Configuration = 3-Way Passive 24dB/Oct
Crossover Type = TP Passive 2nd, 4th Order

Only the **First 3dB** frequency needs to be selected, and additional parameters are R(load), and TP- a-ratio. Figure 9.67 shows such filter with R(load) = 8ohms and a-ratio = 5.0. Please note, that changing R15/R16 you can adjust amplitude for midrange section. Resistor values can be calculated isunf the built-in L-Pad Calculator (see Figure 9.68). Please note, that there is still some contribution from Low and High-pass filters in the midrange frequency range. Therefore the attenuation for the midrange section ususllay need to be selected higher than the SPL hump would indicate.

9.66
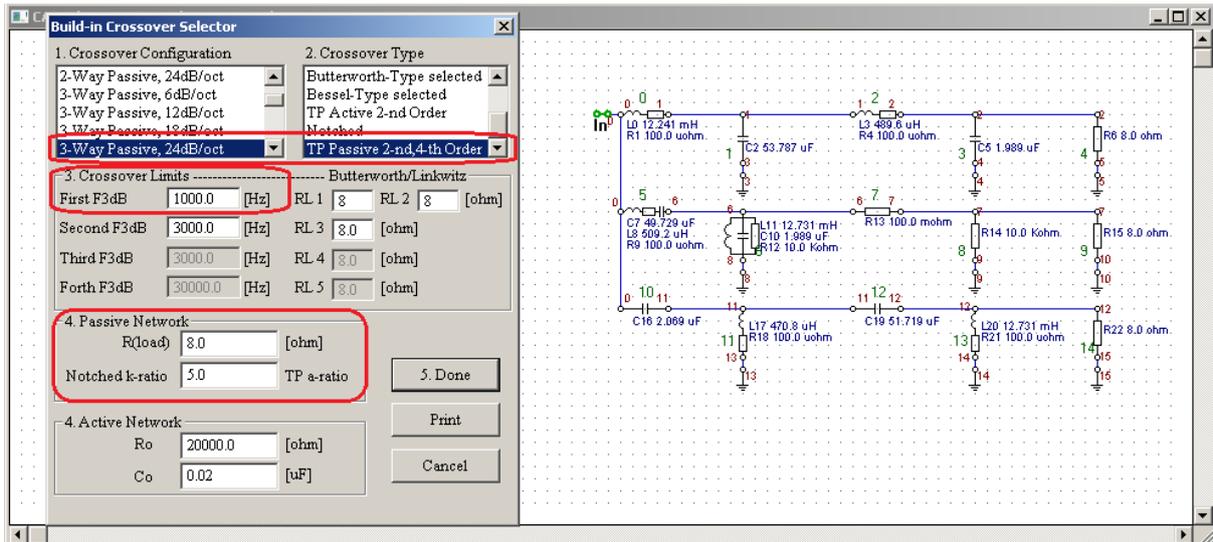
Figure 9.67 Delund Crossover with R15/R16 amplitude adjustement for midrange section.
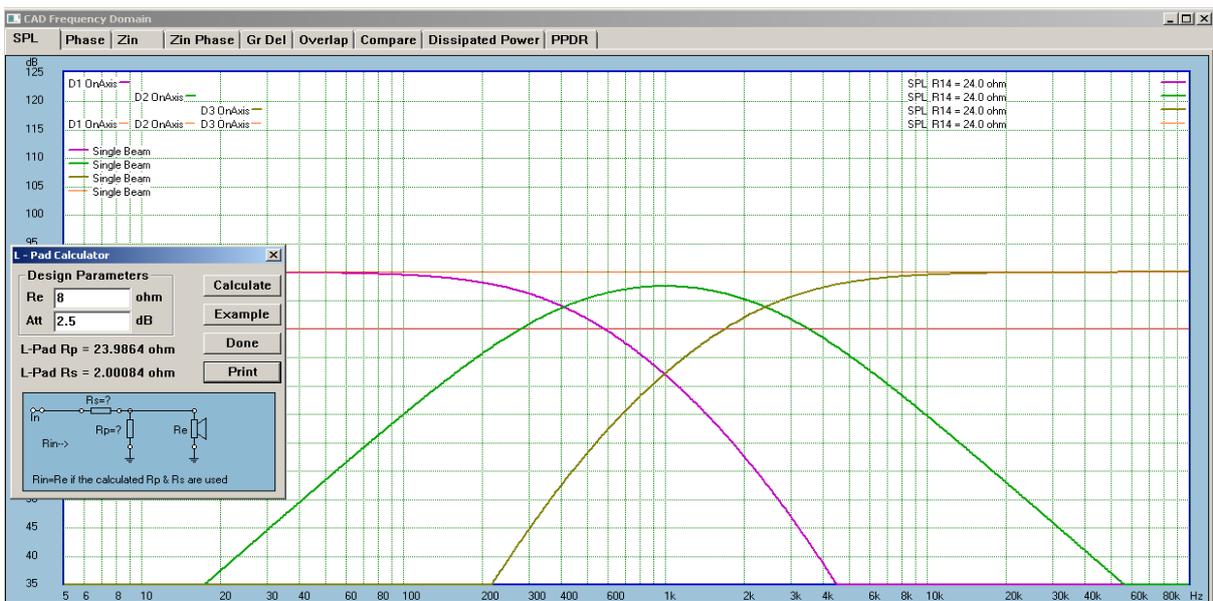
Figure 9.68. Delund 4th-order crossover SPL/Phase plots.

9.67

For comparison, a 4th-order Linkwitz filter is presented below. Please note the difference in phase responses.
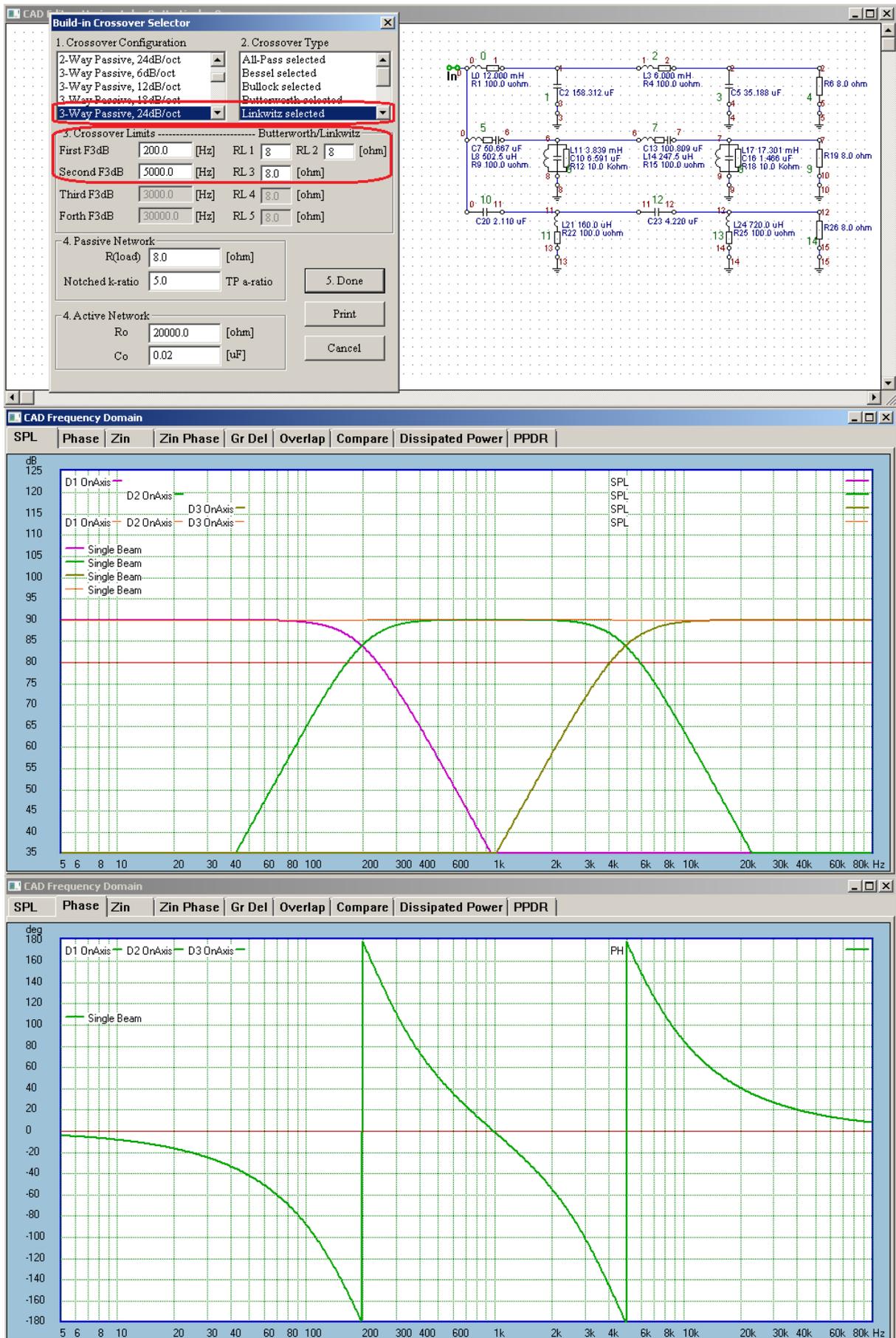
Figure 9.69. Linkwitz 4th-order crossover SPL/Phase plots