Real-Time Digital Signal Processing
Prof. Rathna G.N.
Department of Electrical Engineering
Indian Institute of Science, Bengaluru

Module No # 11
Lecture No # 54
Lab: Echo-Generation

Welcome back to real time digital signal processing lab actually so today we will see on code composer studio and on the board how we are going to run some of the examples what we have looked in MatLab. So the first one I will display is Echo generation.

(Video Start Time: 00:45)

Here it can be either I can take a voice or mic input so we will see how to do the mic input later on if we have a time will give the even the line input what we call it. So whatever the speech I am going to talk about it so you will be hearing it multiple times. So that is what our Echo generation is, so here the buffer says chosen is 4000 and gain for the thing is given pointed. So we can vary the gain so that the intensity is going to be reduced.

So we will be taking the input sample from the mic in and then we will be calling the codec both left and right channel what will be putting it back. So, how we are going to do the thing so we will be having the input and then output getting delayed with respect to gain whatever has been given. So we can buffer length also you can change and then see the repetition how it can be reduced or it can be increased.

So this is after that you will be incrementing your eye sample and then you will be outputting it in the codec. So you will be initializing from the main function here that is mic input what we will be taking it and then running it. So we will see that how the demo is going to run, will because it is recompiled as I have been mentioning in the previous classes. So, if it is done it will be much faster so we will run the thing, hello welcome back to real time digital signal processing course.

So you can observe, how long it is taking for the Echo to come back also. So you will be observing that even if I stop running the debugger mode still the code is running in the board. So I will reset it so you have observed that this is from the mic, so if we change the delay of fetch that is buffer

size we can make it as 2000 and then our gain will try to reduce it to 0.4 or so, and then we will save them and then we will recompile and see how it is going to behave.

We will observe it once again by running it, hello welcome back to real-time digital signal processing course, so you can see that how the echo has got reduced. Earlier it was multiple ones what it was coming because we had a low intensity of the signal that is we said 0.8 and then now it is 0.4 and then only 2000 of the samples what we are repeating it. So this you can go and then play around whatever way you want to do it.

So if you want to run using the line input then I had to give the sample from there I had to comment on this. So we will see running this code I had to because I forgot to save the thing it is asking me whether to save it to this so I can give ok, then it will start compiling and then debugging the thing. So in the meantime we will try to select one of we will select clean voice here either it can be 16 kilohertz or clean voice 8 kilohertz.

So I will choose this as the voice which why I want to run it and then we will see whether how we are going to get the echo from this. So the input signal is coming in, as you can see here there is a mistake in the code that is the reason why you are not getting the output. So what it is selected is line input has to run at 48 kilohertz but the speed signal what I have is at 8 kilohertz. So we will change the input sampling rate that is ADC codec what we are going to select the thing to 8 kilohertz and then we will see whether it is going to run again.

So I had to rebuild the code; this is how you can interchange between the line input and then mic input. So if you want to see your own voice how it is echoing for playing or whatever may be the thing you can give it through the mic input. And then a line input sees that your sampling rate is properly chosen because all speech voices in this case 8000. So if you are selecting any of the music which is sampled at 48 kilohertz or 44.1 kilohertz you can select that and then give it as an input.

So this is one of the thing, we demonstrated in the MatLab so you have seen in the board also how it is running. So we will see the scrambler here but I will be closing this and then we will open a scrambler that is dot C in this case. So what it has is it is going to include the sine 160 dot h and

you have the filter coefficient which is 64 order, we look at the thing and what we have is basically all are interrupt driven here.

So the $x(1)$ will be getting the input from whatever you are playing one that is 160 Hertz sine basically sample. And then you will be combining with your, that is input whatever input speech signal I am going to give it. So either it can be from the mic or same thing what we can select here I have chosen at present mic I can include with it or I can give it as a line input also; both will see the thing here how it is going to happen.

And then what is the thing is going to happen, so y_1 in this case is sine 160 sample that means to say that whatever the input samples are coming you are mixing with the sine function that is you are adding it basically. Then how you are going to do the thing so you are x_2 sample is generated with this y_1 that is get new input into delay line basically. Then this one you would be filtering it that is y n 2 + will be your $h(i)$ so the order of the filter what you have chosen and then multiplied by your input signal $x_2(i)$.

So then what is the thing this again $x_2(i)$ because we have to move it so we will be delaying it here one FIR filter is happening the other place here it is FIR filter what we are doing the thing with both the signals. That means to say that after mixing it so we will be passing it through the filter even the input signal is restricted to or the length whatever low pass filter link. So that we are not passing more frequency in the thing after adding with it we are passing it through a gain filter, so that will be cutting off anything above 3 kilohertz in this case.

So then we will be outputting yn2 to our left sample in this case it is going through the DAC. So what you can do one of the thing is you can repeat this so that again you will be using this as an input scrambled signal is given to the other board if you want to use it, or you can give this is an input and then run this algorithm again then you will be getting the unscrambled output. So we will see now scrambling how it is going to happen first we will be putting it as a mic in.

So I will be doing the debugging is going to happen so it has to it is already compiled so we have to go and then load the code onto the board so it is done. So we will see the output what is you are going to get out of it, so your hearing a one tone that is sine tone hello**.** So you are hello you got

mixed up to give a better feel of the thing what I will do is I will run the, speech signal anyway it is as you can see it is running continuously here my speech signal.

So I will take this is input and then will combine and then see how it is going to look like. So I will call this instead of mic I will be giving it as line, so will compile it will run the code again. So your speech has scrambled so we are seeing in the theory that scrambling is going to keep our original voice whatever we want to convey to the other the one who is interested in not to all the people.

So that at the receiving end, so you know with what this thing sample you have done the scrambling. So they do the descrambling with the same method they can get back the original signal. So I will leave this as a exercise for you to put the code here to run to get back your own voice or your speech signal. So if you want you can record using MatLab your own speech and then give it as an input to this and then see how your speech get distorted so that nobody can make out that who is speaking.

And then at the receiving end you can get back the thing so this is the other example what it was sending. So the next one is what I will do is here it is yCrBr basically what we have it so we will see the source file this is sorry in the previous one. I told you I will show you the filter coefficients so you will be seeing this is a sine waved generation basically 160 which is generated from MatLab and then you are keeping it as a dot h file the other one is your coefficients what you have generated using MatLab again. So you will be seeing that automatically using a function you can use the dsk fir 67 dot m in the MatLab file and then it generates the coefficient order N is equal to 65. Here as you can see that and this is the coefficients what it has been generated either you can use the FDA toolbox or use this code to generate from the book basically.

So now we will see what is our yCrBr so we said in the image processing so I can convert RGB into my chrominance Y and chrominance CR and then BR and then intensity is y elimination basically what we are going to have the thing. So here what it is going to do is as you will see this is from the book co basically that is real time Digital Signal processing to show you that how this is actually this book supports 5 5 0 5 board basically.

So the code how it can be or if it is written in C code how you can reuse in other boards also with little modification what I want to show the thing. So in the DCT part of it as we said it was assembly

code which was lying there so one has to modify because that was meant for this board and then to make it a 6X board you have to modify all the assembly instructions. So you are doing input files are binary formatted so that is files and output file is a BMP file what it is generating.

So only 8 bit data made mode is going to be supported in this and then minimum processing unit is 2 pixels in the same row what it is going to do. So it will be using some of the dot h file here also this is what you have is YCBCR to RGB dot h what you have generated and then kept, so you will be seeing that these are the include files. So in the previous one, we had put all the required files in the same folder here you will be seeing that multiple folders are there.

One is you will be keeping only the source files in this case and then include files are coming in this and you will be having the data in a different place what you will be storing it. So you will be seeing the butterfly structure if you want you can this is getting stored back actually you can rename them and then you will be getting the data in this. So whatever the reference butterfly you can take it and then compare with both of them.

So you will be seeing that this is the dot h file so what you have it so input pointer to Y element so and then CBCR and then this will be pointing to RGB basically. So the width of the image what you will be defining it and then you will be creating the BMP header file with these are the parameters. So this is an initialized 16 bit and then header file what you will have it width and then height has been given.

So you have when you store this image and then you read in the MatLab so you have to take care of these things to read them and then decipher the image then you have to plot it there. So this is types basically what it has been defined so if it is not defined it will be defining it so some of them Boolean and then what are the integer format and unsigned in what you are will be calling it character and other things.

So long what it is defined with un initialized in 32-bit and short will be usually un initialized into 16 and you can have unsigned character is going to be 8 bit what the definition. So you will be seeing that long gain will be in 32 and short will be in 16 and character in 8 so you have to first Define and then use them in your code. So coming to the code we will see the, this thing what we have, I will close the rest of it.

So you will be now what is it your image width chosen is 160 image height is 120 and then you will be calculating some offset so number of rows what you will be calculating is image height divided by 15 what you will have it. And then you will be defining some data hash pragma specifies where your variable data has to lie. So here you call it as data section and it is named as RGB and then you will be getting as image buffer one whatever we are reading it here.

And the other one definition is it is the alignment as it was told that 2 pixels what we will be doing it so it is aligned for 2 bytes basically. And then you will be defining some of the thing and your main variables what it is declared here so you will be opening the file actually that is a data file you have got it that is butterfly 116 into 120 into 8 dot data for reading purpose. So this is the original as your reference, butterfly what you have it, so you will be reading that butterfly the input. And then if it is unable to open you have the error reflecting.

So then again you will be using this is your elimination what you have read the thing y component next you will be reading your CB and then CR for reading purpose, again you will be opening the file and then you will be reading the other data. So then you have to write it back so for this again now what is it CR, one is y the other one is CB and CR. These are the three data what you will be reading from your input file separately.

And then open writing file that is order written is that is what it says it is B 6 and then B G R six are what you will be writing it. So this is BGB dot RGB what sorry BGR dot RGB is the file which is going to be open for writing into the thing. So you will be specifying rest of the sizes in this case and allocating your memory so that you can release it later. So a dynamic allocation of memory what it is happening here and then you will be doing the data one block at a time what you are going to take it.

So you will do the conversion so you are reading from the thing and then do the conversion and then for both your cbcr what you have to do the thing. Then later on you will go back and then write it into initially temporary file and then you will be writing it out so then, once everything is done so you have to free all the memories so that it can be used, so you will be using or releasing them.

So then now if you want you are allocating a memory to your temporary buffer so that is the image width into 3 because RGB what you want to have the thing. So this is the RGB file read you will be doing it and then you will be creating the BMP file here and then write into the thing. All temporary data whatever you have the thing so you will be writing into your output file. So we will see how it is going to run so you will be getting the printed statement continuously.

So we will run the code, so it is doing YCBCR to RGB conversion and writing it into dot BMP. So it it in between you can give the printer statement so how you are proceeding otherwise it may take little longer time to see the thing. So 8 by 8 bit what it is getting operated on as you can see the thing. Working on the row now it is working on the BMP that is writing is happening here.

So now it says that it is completed, so one thing as I have been mentioning the thing since I have did not give for a break point or so what happens is it is unable to find the last line that is what it says. I cannot find a source file at whatever exit dot C it is unable to find the thing so it gives an warning in the end so but we have completed running the complete code. So we can if you want to see the thing so we can go and then tools.

So what we have sorry so what is the one second; if you have declared them as a global variable then you would be able to monitor the output basically. So here you will be seeing that it is declared as Global I can see RGB ok so you can go to tools either I can put the graph, so it will be a variable or I can view the memory basically. Memory browser I can do it and then give my, this thing what is it the thing I will be giving it as RGB here the variable It is declared as INT 16.

So the style because it is declared as INT 16, 16 bit signed did not I will give it so you will be seeing that RGB is the star this is the place where your data will start and these are the values what your storing it in RGB format. This is how it is stored and the other one what we can look at is so if you want to see the offset what is it you can go and then see that also. So the rest of them are not declared global, so if you want to look at the memory of it so declare them Global so that you can see their values.

Otherwise internal to the function so you would not be able to look at it this is one of the ways of looking at it. Now what you can do is it has written I can close it after this you would not be able

to see the memory after running it what you can go and do that. So what it has created is this BGR dot RGB file has got created so this file what you have to do is take it to MatLab and then read it with proper formatting given to the thing.

So this completes the one more equalizer what we have it, so what you can do that also you can take it as an example to run the thing. So you will be seeing that this is a graphic equalizer so it has low pass filter, high pass filter, and then the band pass filter what it has been built across. And then your buffers have been given and then amplitude for that and it is also interrupt driven so you will be getting in low pass filter what you will be passing it through, and then your band pass filter, and then high pass filter, and then band pass filter.

So if you have generated a gel file so you can select just how we selected in MatLab different filters passing through the thing. So in this case the complete thing passes out, so you will be initializing the line input here and then you can generate music whatever you have the thing. And then with proper adjustment of this your band pass filter and whether you want to include all of them as you can see here it is getting multiplied with output from the filters all of them added, so you will be outputting it.

So we will here what is we are going to get it so you have to guess the thing I will be giving the speed signal, so if you want to give it a music you can give it and then look at it. I think my Speech signal is still running so I can we can test what equalizer it is because all of them have mixed together, so you will be hearing the sound there. So what you have to do is you have to modify your gain in this and select whether you want only the low pass filter and then you can eliminate.

So a gel file has to be created for this and then you can select only low pass filter then because we know that up to 8 kilohertz what I can run this code. And you will be seeing that the increase whatever the volume what you have given multiplied by 32,000 what you have it. So I can select a audio file and run it at high input the thing at a high this thing or sampled signal and run this code and then you can check it. So you can experiment with this audio equalizer by only passing through the only low pass filter and rest of it you can comment it out.

(Video End Time: 31:29)

So thank you so this I will leave it as an assignment for you to venture. Thank you for hearing to this lab welcome back and then any queries or anything you can ask me thank you.