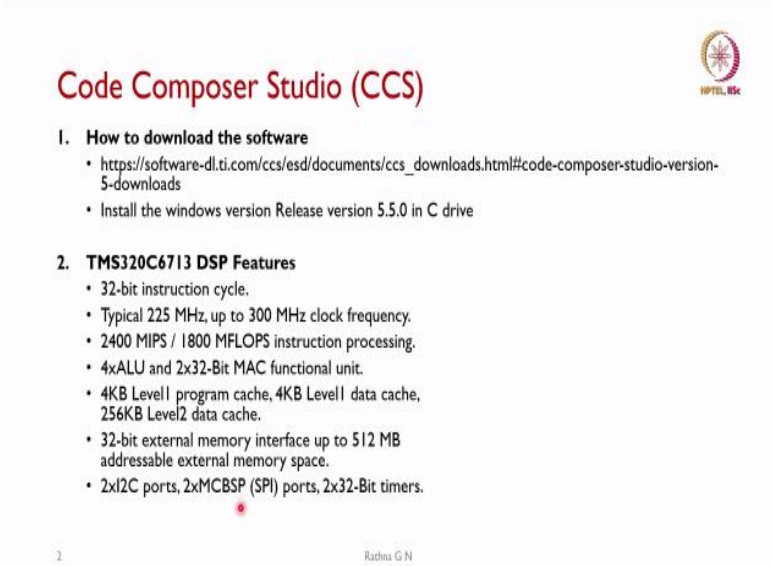
Real - Time Digital Signal Processing
Prof. Rathna G N
Department of Electrical Engineering
Indian Institute of Science - Bengaluru

Lecture - 03
Lab - CCS

Come back to real time digital signal processing course, today, we are going to discuss what are the few things that we can do it in laboratory. So, first lab course is going to be demonstrated today.

(Refer Slide Time: 00:35)



In the present laboratory, we will be using DSK 6713 board from TI. So, in this case, we will see how to go with the code composer studio, and then the first part of it, how to download the software. So the below link gives you how you can download the code composer studio version 5 under the downloads. So this will try to install in windows version, this is the release version of code composer studio 5.5 in C drive, so you have to have a permission to download it.

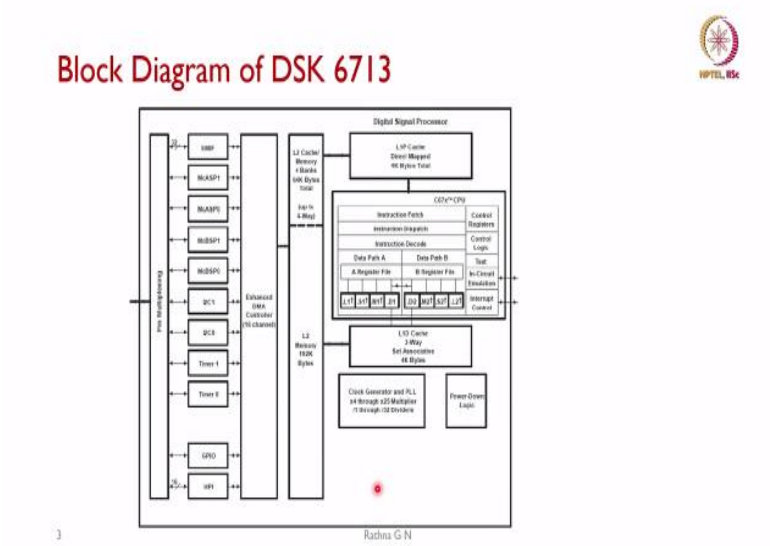
If there is any issue, you can come back to me so I will be giving the license because we have the user licenses here. So you can use that. So first, we will see that after installation, so, what are the features of this DSK 6713 or we call it as TMS320C6713. So, we have a 32 bit instruction cycle processor. Typically transmit 225 megahertz, and it can go up to 300 megahertz clock frequency.

And then what is the MIPS or what we will call it as million instructions per second or mega floating point operations what this board can do is shown here, so 2400 MIPS and then 1800

mega flops instructions processing what it has the capability. So, this has 4ALUs and 2 32 bit MAC functional units, and we have 4KB level 1 program cache and then 4KB level data cache basically, and then 256KB level2 data cache.

And it supports 32 bit external memory interface up to 512 MB addressable external memory space what is provided in this board and we have 2 I2C ports 2 MCBSP that is serial port interface and two 32 bit timers in this board.

(Refer Slide Time: 02:58)



Coming to the block diagram of the DSK 6713. So, we have the core 6713 CPU is shown here. So which has the instruction fetch, instruction dispatch and then instruction decode. So, after that we have the data path for 2 functional units what we have it we call it one is path A the other one is path B. So, it has A registers which are connected to these functional blocks and B registers files are connected to this functional block and this is some of the functional units.

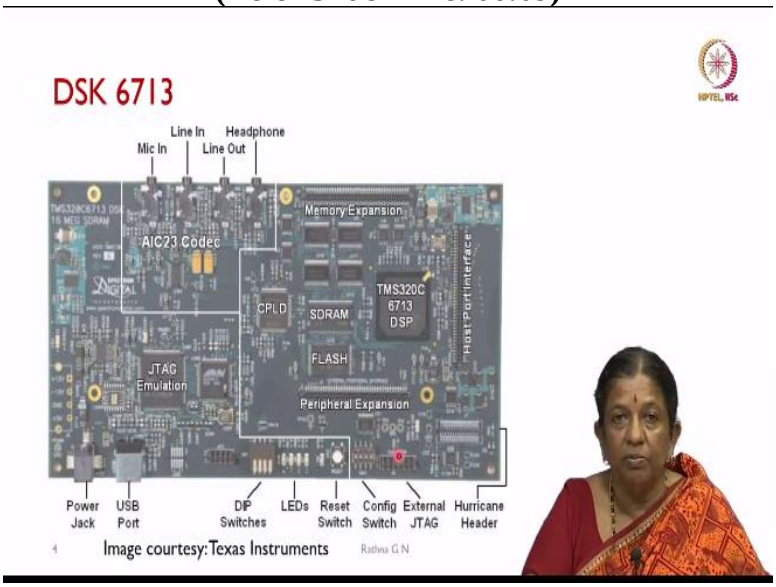
So, when we are taking up in the class about the architecture of the processor, I will be discussing about these functional blocks. Coming to how it is going to be controlled you can see control registers are there, we have controlled some logic and then a test to what we can do the thing and in circuit emulation and then interrupt control is also available. So these are the external which can be connected and you can see that whatever cache memory which is a 2 way set associative 4K bytes L10 is available.

And then we have the clock generator here and then power down logic is available. And then you will be seeing that for this is the data cache what you have seen L1D and then here this is

the program cache. So L1P cache which is direct mapped, in this also has the 4K bytes total basically. And this is the interfaces what we can see this is the L2 memory interface and this is the L which can go up to 4K whatever it supports and then L2 memory which can go up to 192K bytes of data which you can interface.

The other one is we know that whenever direct memory access has to be done we need the DMA. So, we have the external DMA controller here, which is 16 channel what is available and some of the interfaces for the memory interface and then McBSP what it has and then it has a McASP and then there are 2 McBSP ports support logic application specific logic interfaces, I is McASP and then I2C what we saw there were 2 and then there are 2 timers. So, using the timer we can interface the external world and this is the GPIO which is going to be connected to the board and then we can have a host port interface also using the USB.

(Refer Slide Time: 06:09)



So, this gives the overall picture of the board basically. So, we have the DSP processor here TMS320C6713 DSP and then we have the SDRAM, so we can see flash memory is also available and some CPLDs and if we want to have the memory expansion we can go here actually some of the memory expansion and then we have the point external interface for this unit also. And then for the peripheral expansion also what we can do it here.

And then we will be seeing some of the things this is the Mic In for real time application I can use my input as a speech through Mic In on audio stereo what I can put it in the Line In basically from any external device with the audio and then some noise if we can feed in from the Line

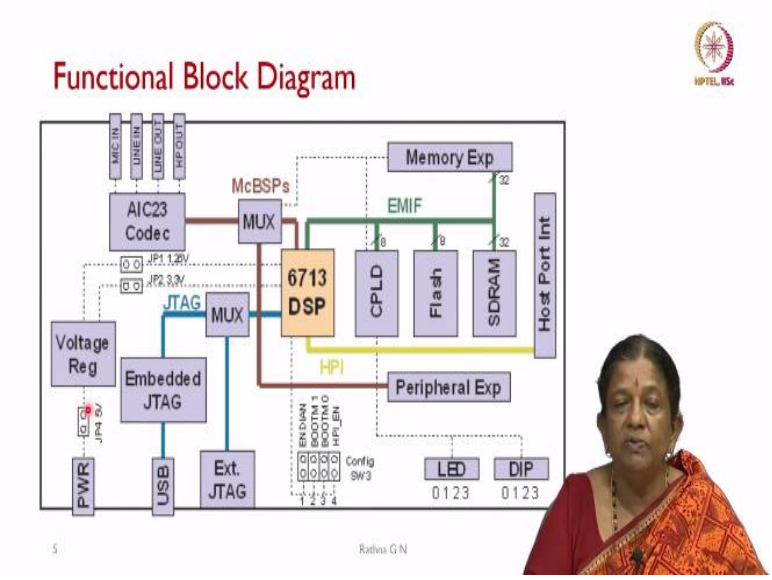
In and then dual output basically here which is Line Out and then headphone. So, both are stereo basically so any from any one of it; what we can take the output.

If a Line Out is taken and then it can be interfaced it with a 1 board as a Line In we can have the loopback system or if I want to hear what is my after processing in the processor, what output I am going to get so I can connect the headphone or speaker out basically. So, we have a little bit of JTAG emulation, so which provides to do the debugging friendly basically and then we have some power jack is what it is going to be connected which is a 5 volt supply.

And this is the USB port which is going to be interfaced with laptop or desktop any of the system. And there are some DIP switches through which you can configure if you want to control your input and output and some LEDs immediately if you want to say whether your board is working or not you can do that and there is a software reset switch available here. And then now there is a configuration switch how this has to boot basically what you can configure with the switches.

And we have the external JTAG interface also so which can be used for more debugging and then go deep into the in circuit debugging part of it. So, this is what they call it as a hurricane header what is available.

(Refer Slide Time: 09:04)



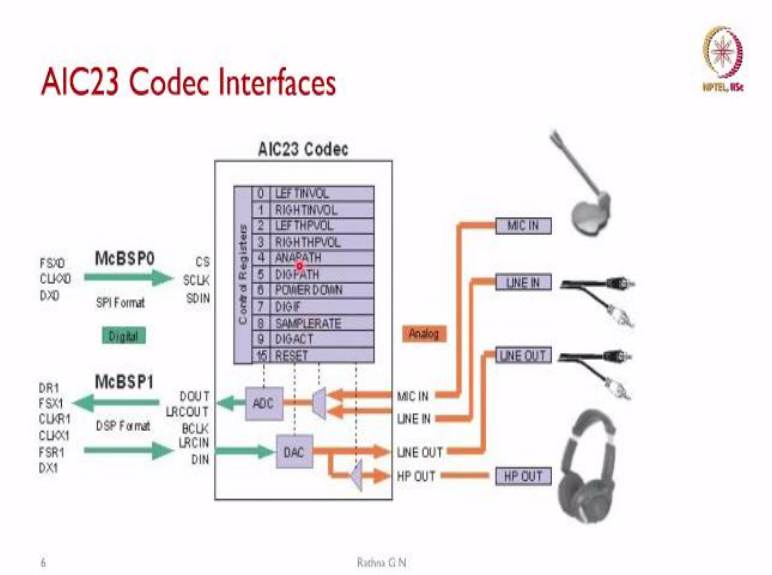
So, coming to the functional block diagram, you saw the board diagram with all the components there from the functional block diagram what you will be seeing this is a chip and then you will have the CPLD logic and some flash memory and then SDRAM and then these are the memory

expansion unit what we have it we call this as the external memory and then we have the peripheral this thing interfaces here expansion for host port interface what we can use the thing.

So using McBSPs one of them is going to be selected either your codec or peripheral expansion, one can used thing. So with the peripheral expansion, so we can add on camera unit daughter card and other things to give extra features to the board. And we have as I said; in the theory class we will be using the AIC23 codec in this board. And other higher versions will be using AIC23 one versions. And as we saw in the board, these are the inputs, 2 inputs and 2 outputs what is available.

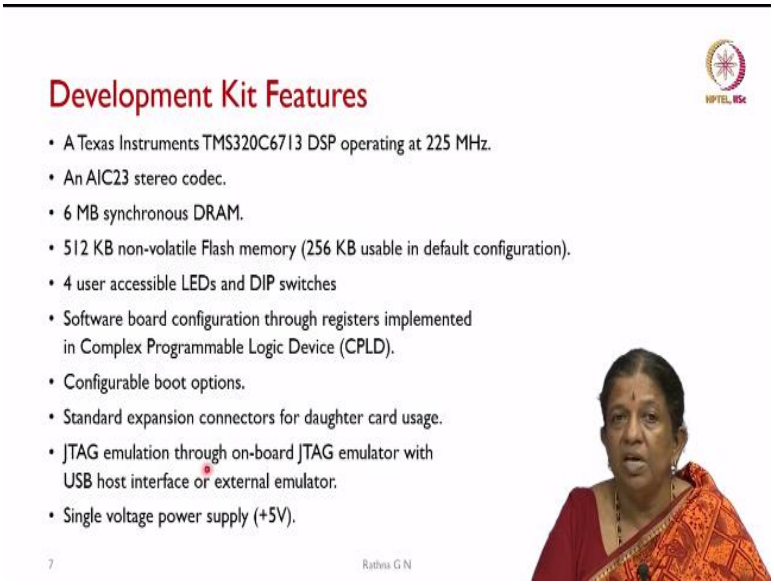
And then we have the JTAG interface using the USB we will be connecting and voltage regulator. So here it is going to be 5 volts, what will be operating some of the boards 5 5 series will be operating at 3 volts suboptimal voltage to run the board.

(Refer Slide Time: 10:41)



So we will see little bit on AIC23 codec interfaces, how it can be done. So we will be using the McBSP0 to input to this codec chip and then ADC output is going to be taken out through McBSP1 interface to the DSP board and then whatever the format it has to be defined, we are going to provide it here. And then as we can see that I can given as a Mic In, or stereo we will be seeing input or stereo output, or I can connect the headset for it. And then we have different configuration here. So we will be seeing in later classes, how we will be interfacing these things.

(Refer Slide Time: 11:32)

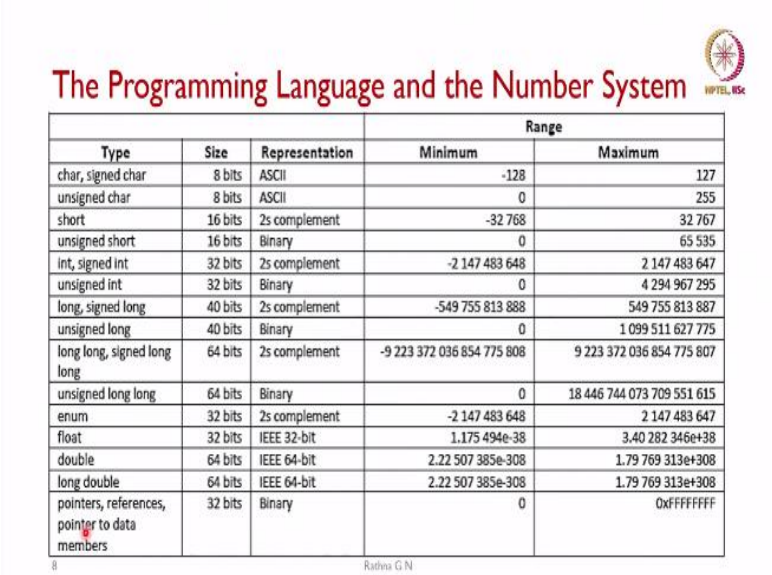


So coming to a development kit features in this case. So we say that this is Texas Instrument, TMS320C6713 DSP, operating at 225 megahertz. So it has AIC23 stereo codec, and then we have 6 MB synchronous DRAM this is complete to the board what we are using it or whatever features I have mentioned, it will be referring to the board, here we can go up to the disk memory if it is we in the full fledged board basically, this is a development kit, what we will be using it which is a subset of this features.

And we have 512 KB non volatile flash memory which is 256 KB usable in default configuration. And we have 4 new user accessible LEDs and DIP switches which can be configured for depending on our application. And we have the software board configuration through registers implemented in complex programmable logic device that is the CPLD and we have the configurable boot options. Once the system is booted it can run on its own once the power is given.

So it can continuously take the input and then it will be providing the output only for loading your program and then debugging purposes we should have the interface with one of the hardware either laptop or desktops and then we have the standard expansion connectors for daughter card usage if it has to be connected, and we have a JTAG emulation through onboard JTAG emulator with USB host interface or external emulator and as it was said that it works on 5 volts power supply in this development kit.

(Refer Slide Time: 13:36)



So coming to the programming language and little on the number system detail of it we will be taking it up in the theory class. So, the type was DSK 6713 board supports and CC basically not complete C is available. So based on that we have the characters and signed character are going to be 8 bits and the representation is the ASCII format and minimum is -128 and then maximum what it can go up to is 127.

And we can define unsigned character, which is going to be 8 bits there is also in the ASCII representation which has the representation and the minimum from 0 to 255 what it can take the range, this is short, which is 16 bits so which is represented as 2's complement, so which will be having the range from -32768 to 32767 and unsigned short is going to be 16 bits but it is going to be binary, so it will vary from 0 to 65535.

Coming to integer and then signed integer it is represented as 32 bits and then the representation is going to be 2's complement. So, the range what you can see which can take in the board is so much for 32 bit. Coming to unsigned int, it can take 32 bits binary it will be from 0 to 2^{-32} to -1 is the value of what it is represented here and then even it supports long unsigned long here it is going to be restricted to 40 bits.

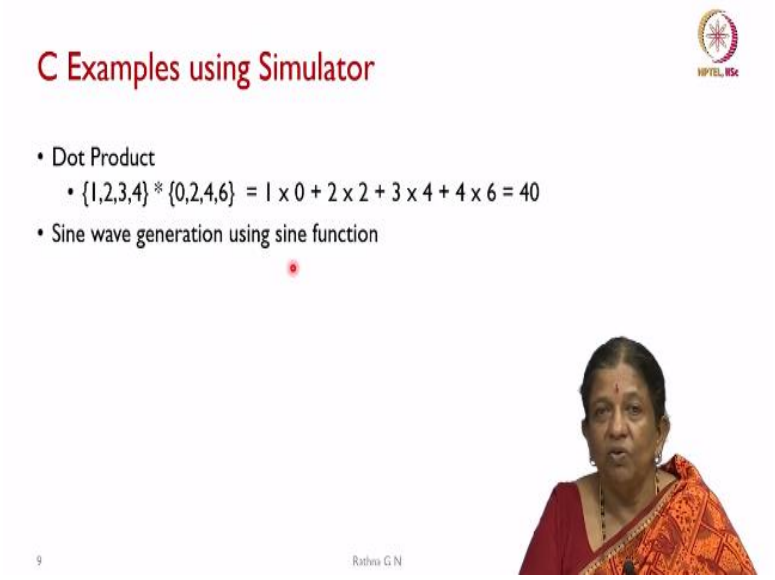
So, you will be seeing that usually long and then sign long has to be 64 bit, but only 40 bits is supported in DSK 6713 board. So, if we want to use 64 bits, then we have to provide long long unsigned long long so, which is represented as 2's complement and then the range what you will be seeing 2^{-40} to $2^{-40} - 1$ and in this case $2^{-64}$to $2^{-64} - 1$ on the positive range. And

this thing unsigned you can have long long which is going to be 64 bits which can be binary. So, 0 to$2^{-64} - 1$ will be the value in this case.

And then enumeration what we can have so, which is also this thing 2's complement and since this is a DSK 6713 is a floating point board this is going to support float 32 bits and double is going to be 64 bits. So, the format what it is going to take is IEEE 32 bit format and for 64 bit is IEEE 64 bit format and then the range is going to be as specified here. And then a long double also what it supports it is 64 bits and IEEE 64 bits standard what it will be using and that range what it is shown here.

And like any other processes you should have pointers, references, pointer to data members. So, which everything is going to be 32 bits in this which is a binary format and 0 to as you will be seeing it is in hex in 0x all FFFFFFFF basically.

(Refer Slide Time: 17:29)



So we will now we will go to some of the example we will show 1, I will be showing sine wave generation using the sine function on the board and then the dot product will be working on the simulator, both can run on the board, but to show that those who are unable to get hold of the DSK board, some of the examples of whatever are shown here can be run using the simulator.

So, as an example, for dot product, what we have is we have taken one of the x(n) is 1 2 3 4 4 values, and then the other one is 0 2 4 6. As we know that the dot product of these 2 numbers is going to be given by this formula. So, the total what we are supposed to get is 40 and that sine wave since we have discussed in the theory class that what will be the case if there is

aliasing happens if proper sampling frequency is not selected. That also we will be showing you that what is the result of it. So we will go to the board to run our examples.

(Video Starts: 18:49)

So I will try to open the code composer studio as you can see, there are 2 versions of it in my system. So this is a code composer studio 5. So you will be seeing the icon is going to be represented this way and if it is higher versions, more than 6 or something like that now 11 version is also available so it will be having this but for this interface, you have to have a board connected to that. So to show that the simulator is going to run only if the simulator is provided up to version 5.

So we will click on the code composer studio. So it will show you what the workspace you want to select is. So wherever your workspace either in C drive or D drive or whatever multiple drives you have it you can use that directory in this case what I have used is so D colon C6713. What I have given the name and then we will try to open this workspace. If it is not there, if you are restarting it you can now browse where you want to create this directory and then give okay. So it will take a little time to open it.

So if any of these things upgradation is going to happen, so it will be showing that what are the files available for you to upgrade. So better not go for the upgradation because sometimes compilers get corrupted so you cancel them. And then it shows Welcome to code composer studio V5. So if you want to create a new project, so we can open here, so I will show from the scratch although I have 2 of them.

So I will show you a new project will open it. So it will show you what is the requirement of the thing project name what I have to give, since I have already have the dot product, what I will give is dot product 1 I will give it so that I will be creating a new one. So then it what I want output type is executable which is default. And then the processor families you will have different processor families, it depends on how we have now installed your code composer studio.

Since I have board for with arm and then of 54x series also boards we have in the lab and 55x board are also there, I have given option of this. So if you do not want anything you can give only C6000 option in your case and then go with it. So in this case, I will be selecting platform is C6000. Because 6713 is the board what I will be using it and then I have to select what kind

of this thing or type of filter if you want to have the thing or directly you can select it as this the DSK 6713.

Now here, what I have to do is whether I want to have that simulator or the emulator. If I am connecting the board, then I have to select that spectrum digital is a DSP onboard USB emulator if I am connecting the board, so if I am running in the simulator mode, then I will be selecting simulator. So, first we are doing the dot product as I said, I promised that I will be showing you in the simulator mode. So I will be selecting the simulator in this case and then we can do finish.

So, you will be seeing on the left hand side of what we are the new project has got created. So and then you will be seeing that there is a C basic minimum has been coming here. So that is in return of your main is integer what you are calling it. So what and then return 0 what it is going to be the default with braces what you will be seeing it. So now, there is a little this thing, what it asks for is the target configuration.

So when I go to this one, and then click on this board, so here you will be seeing that that will be a little emulator what it will be coming. So what I want is the little Indian. So one thing, what we say is a little Indian means that in the memory lower bits are stored first and then higher bits are stored later on. So when it comes to big Indian, then it is going to be higher bits first and then lower bits later some interfaces may need that higher Indian, but most of us need we are comfortable with lower bits first and then higher bits later.

So we will be using the little Indian in this case, and then it will be asking default there started with the 62x that is why it will be showing 62x as the thing and some gel file wherever it has downloaded it is showing me emulator here. But I want to have a simulator connected to this. So what I will do is I can first thing is I can remove this CCXML file, and then create my own new one. So we will see that how to create my configuration file. I can give delete, and then delete it.

And then now I will go to right click on this dot product 1. And then I will say new, what I want is the target configuration file. So I will say yes to this and then I will click on the new configuration file. So you will be seeing that it is Texas instrument simulated what it is giving

me the thing. So now what I will do is I have to come back to my main dot c, and then put the dot product.
So since I have already created the dot product, so we are calling it as a short, x count is 4 numbers what I showed in this thing slide and then y count is going to be this. And then what we will do is to save the time, we will cut and paste this in our new directory. So I will be replacing this also main also because everything is given there. So this is a default main and we are initializing result to 0 and then result is what we calling as a function dot product of x, y, count.

And then finally, we can print a result because this is an integer what we have taken the thing short is 16 bit as already format, what is specified it so I will be printing the result value in the integer format. So then the function dot product as you can see the thing, so you will be initializing i value and sum is 0, then depending on the count in this case count is going to be 4 what it is defined. So it will be less than count.

So we will be incrementing it and then you are doing the product of a(i) and b(i) and it will be returning sum to the main this thing function. So what we will do now is you are seeing so many blocks on the thing. So this is where your project explorer is going to lie. And then this is where your C code is going to be seen. If you are writing in C and if you want somebody wants to do assembly, we can do that also but the extension is going to be dot asn.

Here, as you are seeing it is main dot c, so we can interface assembly and then C programming if time permits, we will be seeing one example of that also in the end. So now what I have is a console and then if there is any error what it will be coming and this is GUI basically a code composer studio. So you have different options in this case, so we will be making use of one by one.

In this case what it shows is either there are icons to do build debug and then compile the thing or I can go to the project option and then I can say that build automatically what it is shown, so I can use this or this one to see whether I have any errors in my code basically. So what it says is its telling me reasons error in the case. So I can click on it to show me that so that is undefined main dot c what it is showing me here.

Here as you can see count is undefined. So we will define the count here I can call this also short is equal to 4 what I will give the thing and then let us see by compiling again. So if you have not given control s so when you are trying to compile it will ask you whether to save the thing. So as you can see the thing, I forgot to include these 2 in the beginning. So because for printing I need to have these 2 function defined.

So we will be defining the count here itself so I can remove the count from here this place and then now we will try to compile the thing, so as you can see the error is gone now, but still I have a warning. So what we will be seeing sometimes these warnings become critical in the board. So in the simulator, it does not matter, but we have to take care of this warning also. So what it does is it is showing is stack is not defined and then some system memory has to be defined for heap operations.

So what we can do is right click on the thing and then go to the properties and then we will be getting the properties for this project. So here we have the compiler option because it is the linker option so what we will be doing is basic options what we have to take it, so you will be seeing that stack whatever the size has to be defined here. So I will give it as a 512 as the stack size and then you will be seeing dynamic memory allocation that is basically heap size. So you can give it as 512 in this case also and then do okay and then do the recompilation.

So you can see that there are no warnings also now what I can do is I can go to build and then I can build a working set whatever doing the current working set or I can I will be giving the or build this as you can see in the thing dot product, so you will be seeing that this is the active directory so I can keep active debug also or there is an icon so that is a debug this one because this is the current one I can do debug now what happens is. So, in this case what I have to select is connection type what it is asking so I have to select the simulator.

So one way of doing it is I can copy whatever I have said the thing into this and then paste the configuration file and then I can go for debug, here because it takes little time to set the simulator, what you have to do the thing. So what I will put is how to set a breakpoint, because when I run the thing, I do not want it to go beyond my control. So what I do is if I double click on whichever line I have it, so it will be generating a breakpoint here.

So now, either I can go and run in different, I can load the project and then I can resume from wherever I have stopped it or I can terminate or I can software reset what I can provide from the code composer studio or I can restart or I can do step function in running or step over that particular function and then go to the next one or if you have done assembly programming, so you I can go into the assembly thing also.

So here, what I will give is that there is an icon to show that, so I can click on it. So it will be running, as you will be seeing that, from the start of the thing, it has run up to the deep wherever I have put the breakpoint. So now you will be seeing that the sum value is 40. So this is how the product has been calculated. So I can stop debugging this and then what I will be showing one more example on the board.

So you will be seeing that sine generation using what I will do is I will close all of them because it may be getting confusion for you people. So I will click on the main dot c here. So I will be using the math dot h and then we are defining the sampling frequency initially we will define it as 8000. And then π value has to be defined here it is not, some of them are not explicit, so we have to provide it to the system.

And then we will be defining the float as my output. So the length of it, I have given it as 256 depending on your sampling frequency, and then sample period, how many sample periods you have to generate, you can specify it, and then we are starting the main. So the frequency of interest in this case, what I have given is 1000 Hertz. And if I want to have the magnitude of it or amplitude of it to be increased, I can give that value and then multiply the all the output by this value and then I will be enhancing it.

And then I will be calling θ increment as the value of sine function. And initially, we are going to define θ as 0.0. And then this is going to go for a loop up to 0 to 255. So what is it? So I will be incrementing. So as I said, I will be using the sine function down actually. So, we are defining $2 * \pi$ into frequency divided by in my sampling frequency, what I will be giving as frequency of interest and this is the sampling frequency.

So, in this case 1000 / 8000, what it is going to be 1 / 8 will be the value of this and θ will be incrementing it. It is going to go in steps plus equal to and then if it goes beyond 2π will be

resetting it to 2π basically. And then output of i what we are going to calculate so amplitude to increase its amplitude I said and we will be finding the $\sin\theta$. So, we have different ways of sine generation.

So, we are using the built in function which is defined in math dot h sine function basically, and that will be calculating this. So now what I will do is since it is already precompiled, so you can see that it has compiled and here you will be seeing a little change in the thing, I will show you the configuration thing here. So you will be seeing that it will be using the emulator basically that is DSK spectrum digital is DSK board what it is going to use so we will do the compilation.

And you will be seeing that when it is loading onto the board you saw that green one running so that the complete code and then go it will be placed on the DSP processor basically and then it is ready for running. So you will be seeing the pointer here. So here also to save the thing I have given the breakpoint so that we can have the control of it. So it will not be going back to original place or it may go anywhere else also.

So, we have a random thing. So, you will be seeing that whatever the values that has modified the previous value what it was and then after the initialization you will be seeing that some of them are in hex actually, and some of them are in the here you are seeing the integer value the same thing where the location they have been stored, these variables have been shown. Now what we will do is one of the other functions what we will be looking at it.

Because if I come out of it, this tools is not going to be visible, either I can have a memory math file or I can see whatever value stored in the thing I can save the memory or before running it I want to load into memory some values, I can do that or with some pattern I want to fill the memory I can do that. And then real time operating system what we can use the thing and system analysis tool box is also there somewhere if I want to use the hardware trace analyzer, we can see that and now I am interested because I have generated a sine wave.

So how does it look like either a sine wave or not what I want to check the thing so, since I have this is a graph, what am function what it is available in the code composer studio, so we will be using that and here because I have taken it as a floating point. So I will be defining my

thing in data type is 32 bit floating point depending on whatever you want to define it, you can do that. So now sampling rate is I have used is 8000.

And then the start address, what I want is I want to see the output here, if you want to see any variable you can give the name of it, it will be showing you what it looks like. So, you will see that the sine wave has been generated here. Now I want to know at what frequency the sine wave has just got generated. So what I can do is again, I can go to tools, and then here now graph, I have an FFT magnitude here. So we will be taking the theory also, but sometimes to see that what frequency I can use a built in function also.

So here I will be putting it as 256 and then this is 32 bit floating point what we have it and then the sampling frequency at present is 8000 and then start address is output what I want to see, so, and then here also if you want to have a more frame size you can choose the thing FFT order chosen in this case is 5 which is 2 power 5 is 32 is the frame size. So, if I want to increase it to 8 it will be becoming 256 you can check the thing and window function what it is using at present is a rectangular window.

So, if you want to modify these functions, you can do that. So, you will be seeing that what we generated frequency was 1000 hertz. So, you will be seeing the peak coming at 1000 hertz. So, if you want to see you can move this cursor here and then exactly point to the centre of it and you will be getting the exact value of the frequency what you have generated. So now I will show you what is the thing is going to happen to our reconstruction because this is reconstructed.

If I change the sampling frequency, for a simple case I will take it as 800 Hertz is the sampling frequency what I am choosing, whereas my highest frequency component is 1000. So you may be guessing what is the output I am supposed to get? So hold on for a while and then we will see whether our theory and that practical goes hand in hand so this is again, I am recompiling and then loading directly onto the processor. I am going to run the thing.

So now we will see that go back to the tools again, I will go to the graph. So I will see the single time graph again, both of it what we will be getting it so this is a 32 bit floating point what to have the thing and then now the sampling frequency what I have used is 800. And start

address for me again, I want to see my output. What is the thing I will be getting it out? So you will be seeing the compared to the previous one, how the input frequency has been created.

Now we will see it is FFT magnitude what output it has created? Coming to the graph, we will do FFT magnitude, again here it is 256 and then this is 32 bit floating point and then sampling rate is 800 and then this is output. So, if I want to change the thing I can make it 6. So, you will be seeing that it becomes 64. Now, you will see that is 6 matching with the theory, I am supposed to get reconstructed signal as 200 hertz. So, you are getting 1000 minus 800. So, you are seeing that it is earliest version of the 1000 hertz is going to be 200 hertz.

So, this is how you can play around with your simulator or connecting the board. Even this code runs in the simulator one can check it and then you can get acquainted with the simulator or the board so that more and more labs are going to be run based on it. So, if you are comfortable in running few of the experiments, it becomes easy for you to fill around with so many other real time interfaces and others things which will be taking it the next laboratory session. Thank you.

(Video Ends: 46:21)