Introduction to Embedded System Design
Professor. Dhananjay V. Gadre
Netaji Subash University of Technology
Professor. Badri Subudhi
Indian Institute of Technology, Jammu
Lecture No. 14
Programming Methods for MSP430

Hello, welcome to a new session for this online course on Introduction to Embedded System Design. As usual, I am your instructor Dhananjay Gadre. Today, we are going to look at various ways of programming the MSP430 microcontroller. In the previous lecture we have seen the topology that we have that the hardware access that we have. Today we are going to see how we write our programs and how to download them into the memory of the evaluation kit which is the MSP430 lunchbox.

(Refer Slide Time: 00:58)

Now there are many to software tools, software development tools that are available for MSP430. The most important of them is the Code Composer Studio IDE which means Integrated Development Environment provided by Texas Instruments. We are in fact going to use this in our course here.

Other options are that we can have an Energia, Integrated Development Environment, environment and Energia gives you a look and feel as if you are programming MSP430

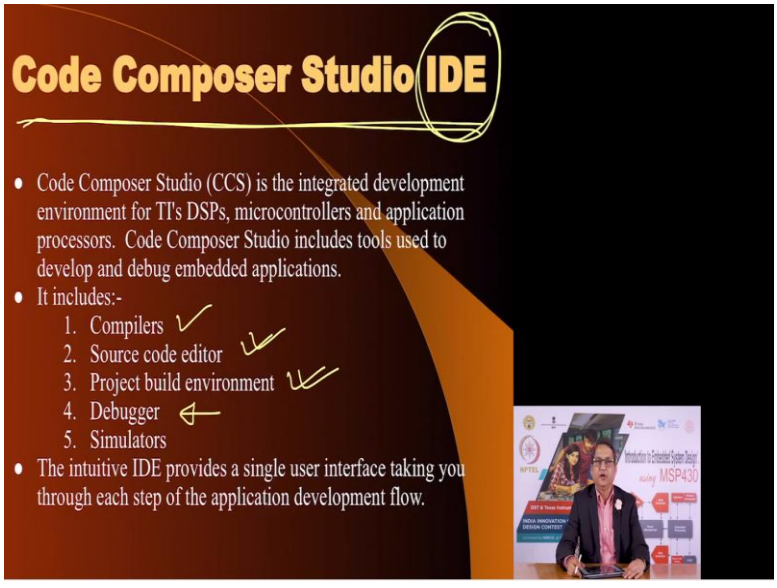
microcontroller like an Arduino and platform. You are free to use that if you like outside the, this course.

But for this course we are not going to involve ourselves in Energia because it keep wrap provides a wrapper layer, wrapper around the actual code which is running on the MSP430 and distances you as a designer from the hardware feature of a microcontroller. And so it is good for beginners. If you are in a school or beginning such activities in college Energia would be a good environment.

But since we are training ourselves to be good embedded design engineers, we would try to work as close to the hardware and the peripherals of the microcontroller as possible. And therefore we choose to use programming languages such as C, or maybe assembly language if you are interested in.

So, Energia I mentioned as one of the options that are available. If you develop applications in a professional environment, perhaps you may use third party tools. And one of the very popular third party development environments is the IAR compiler. And besides that, there are some GCC GCC stands GNU C compiler. There are some open source packages also available which can target the MSP430 microcontroller. These are the options that we have.

(Refer Slide Time: 03:01)



Now Code Composer Studio as it is called by Texas Instruments. This is a complete integrated development environment when I say Integrated Development Environment means that you do not have to invoke various software's for writing your code and compiling them and debugging them and all that you get a single window and through which you can do all the activities required for the editing, debugging, compilation, simulating, downloading all those things you can do through a single window and that is the meaning of this term integrated development environment.

So, Texas Instruments code composer studio offers you that opportunity. It has access to a compiler. It offers your editor which means you can write your program, write in the single window interface. You can build the project environment with various options that it offers. Once you have compiled your program, you can debug it, you can test the code, you can simulate it, you can connect it to various simulators.

This single window allows you to do all these things without having to switch windows and copy paste or interface or import files from other software's. So this is a very good thing. In a subsequent lecture, we will show how to download the code composer studio and get it to work on your development platform which could be a desktop computer or laptop.

(Refer Slide Time: 04:41)

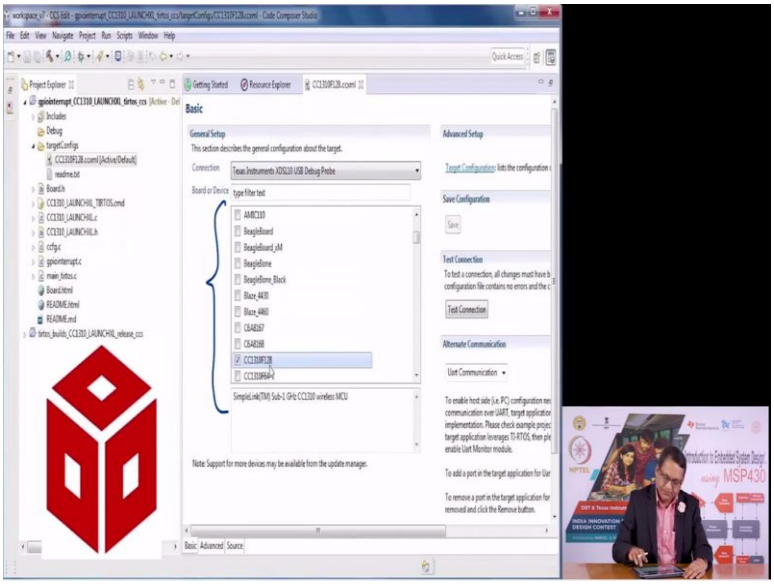
The Code Composer Studio is based on Eclipse software framework. Eclipse is a very popular framework, which allows designers and embedded engineers which work on multiple platforms

they can use the eclipse framework for the development activities. It is supported on Mac operating system. Or you could be using a Windows machine as well as Linux. And it is supported from Texas Instruments on their support forum called E2E.

The website for this is e2e.ti.com. If you go to this website, you will see a lot of options that you have and you can choose an appropriate forum to raise any questions or doubts if you recall, in one of the previous lectures, I had mentioned, that choosing a microcontroller, you should consider a possibility that the microcontroller that you choose has some sort of support mechanism.

And so, here it is the right point to point out that Texas Instruments has fantastic support forum through this e2e website, where various issues related to the development of your circuit or your product or your software could be raised and the community which could be employees of Texas Instruments but also educators and other freelance workers could be engaging themselves in that discussion and their views and opinions could be off value in solving your problems.

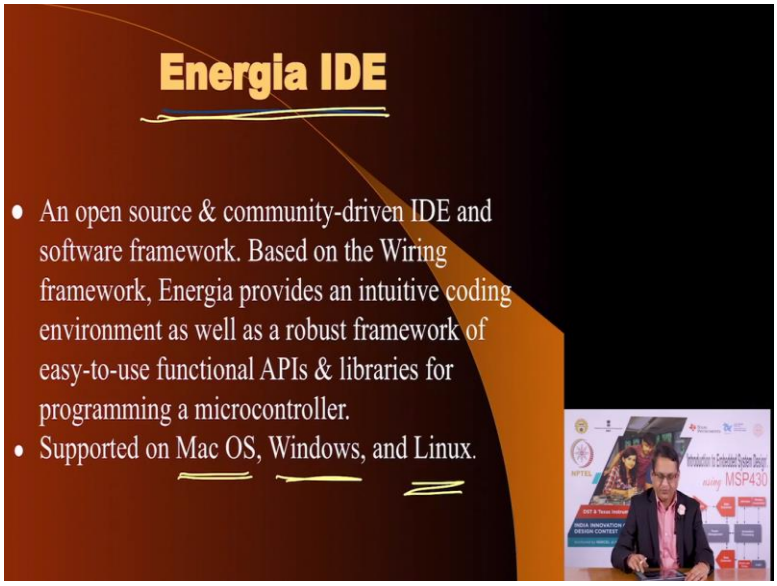
(Refer Slide Time: 06:18)



This is the window of the code composer studio ID that would appear on your laptop or desktop once you install it. On the left you have the various ways of open you to open your projects. The middle part is where you select as you can see, it offers you multiple platforms to choose from and in our case we are going to use MSP430 which does not seem to be visible in this but trust

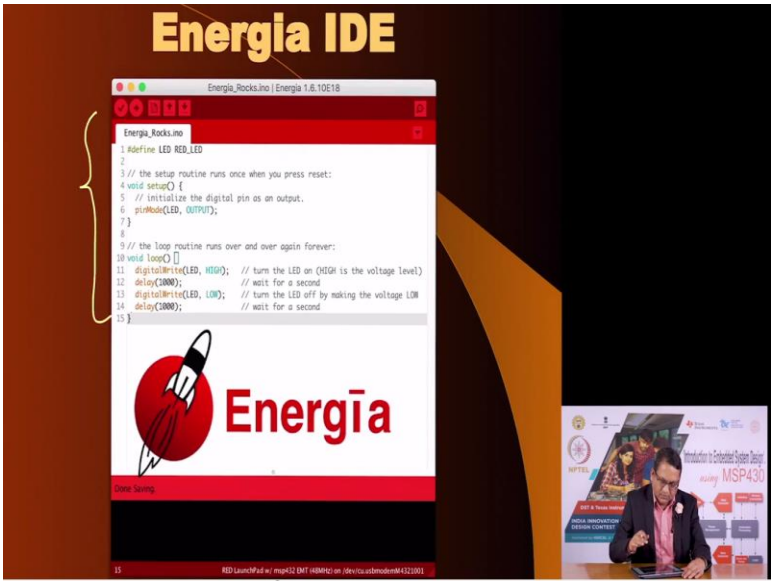
me that code composer studio supports MSP430. And on your right it allows you to enter into various setup environments.

(Refer Slide Time: 06:58)



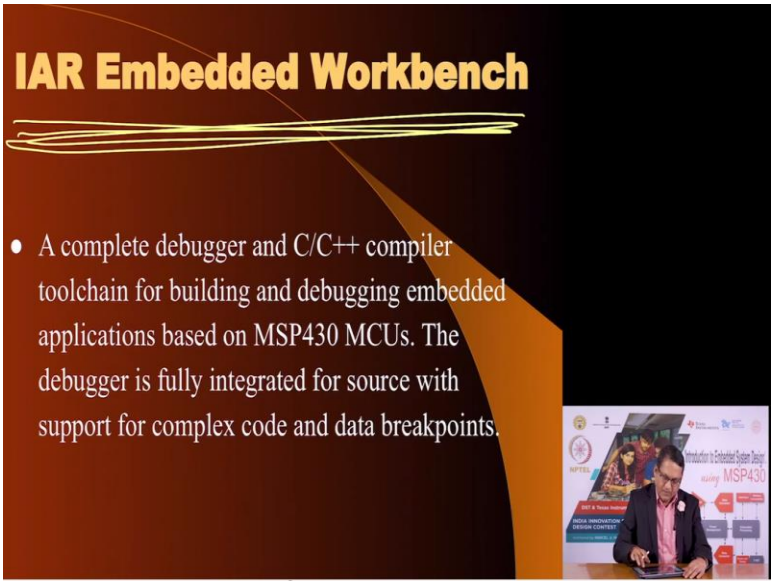
Apart from Code Composer Studio, you could also if you like, experiment with the Energia IDE. This is an open source community driven, integrated development environment just like Arduino. And you could think of this as, as Arduino for MSP430. And it is also supported on like the Code Composer Studio. It is supported on Mac OS, Windows as well as Linux operating system.

(Refer Slide Time: 07:27)



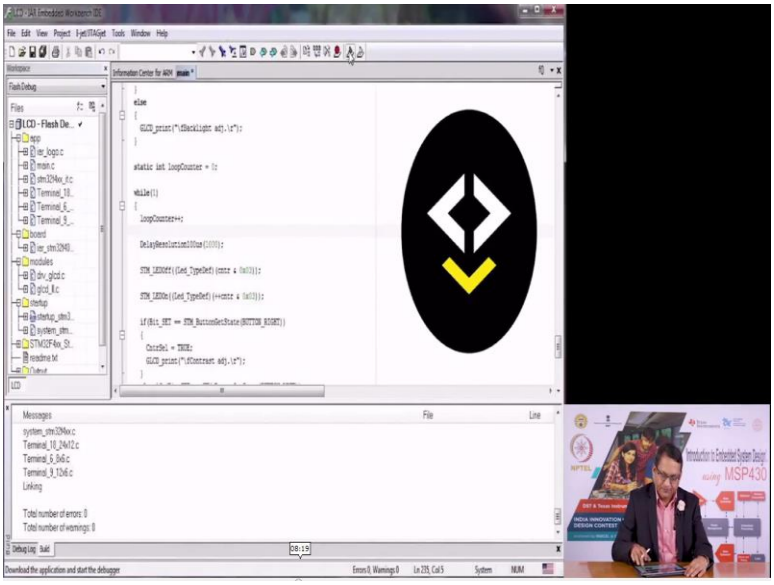
This is what Energia IDE looks like and perhaps this code examples here. You may if you have experimented with before with Arduino, you can see that these are very similar to the coding style and coding commands that you use during Arduino development.

(Refer Slide Time: 07:51)



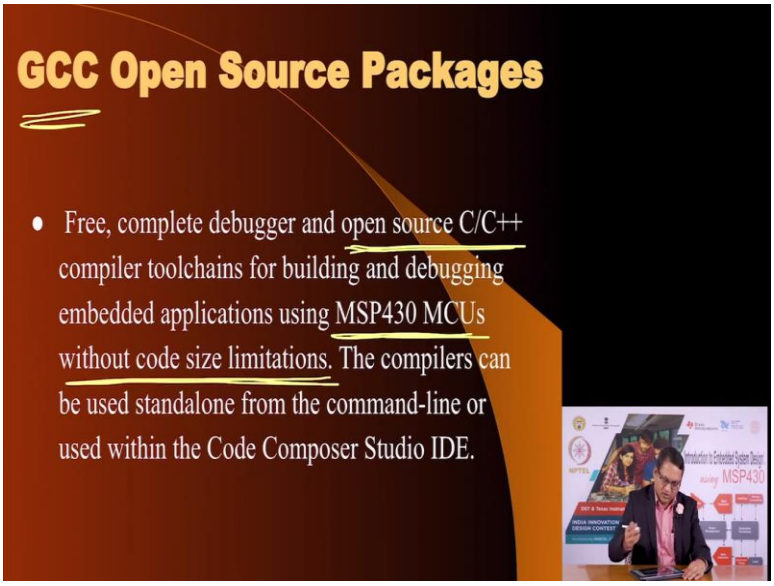
Then apart from that you could engage with IAR and I am very sure that IAR offers you an evaluation version where you can try it out, download it and try it out if you wish to wish to use it in the context of this course.

(Refer Slide Time: 08:09)



This is what the IAR bench workbench looks like. And as you can see this has a look and feel just like the Code Composer Studio.

(Refer Slide Time: 08:18)



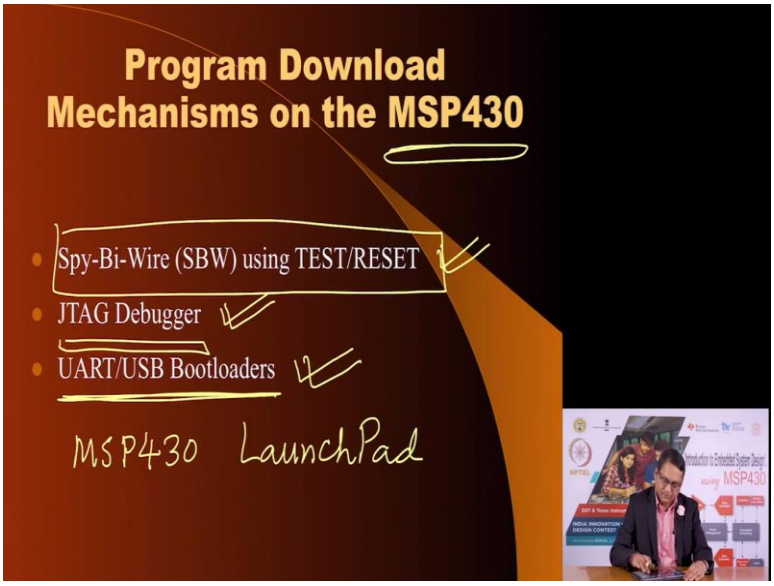
Apart from all these options, you can also use GCC mechanisms which is using open source packages, which is a free complete debugger and open source C, C++ compiler tool chains for building and debugging entire applications from scratch to finish around the MSP430 microcontroller. Oftentimes, this works on a command line. And one of the advantages of using the GCC tool chain is that it does not have any code size limitations.

Many of the up evaluation versions would put a limit on the maximum code size, maximum size of code that you can experiment with. It just to give you a feel that, this is how my compiler is, you if you like it then you can go ahead with the purchase or something like that, GCC does not have those restrictions, and you can choose to use GCC for your applications.

Now, once we once you have chosen to use a particular environment for your application, then we would in the subsequent lectures will show you how you start writing your code and how you compile it and after you have compiled it, look at the, the messages that compiler such a Integrated Development Environment generates to understand how it has compiled the code, what is the size of the code?

If there are any warnings, how to interpret those warnings if there are any errors of course, you cannot proceed so you must go back to your editing part and fix those errors. But once you have done all that, with whichever approach you take, you would be ready to download that binary code or the hex code into the memory of the microcontroller.

(Refer Slide Time: 10:18)



In this case, the MSP430 and we have 3 options. You could use the Spy Spy-Bi-Wire method, which uses the test and reset pins on MSP430 microcontroller. Or you can, you could use a JTAG Debugger. And you could use the poor man's alternative by using a UART USB based Bootloaders.

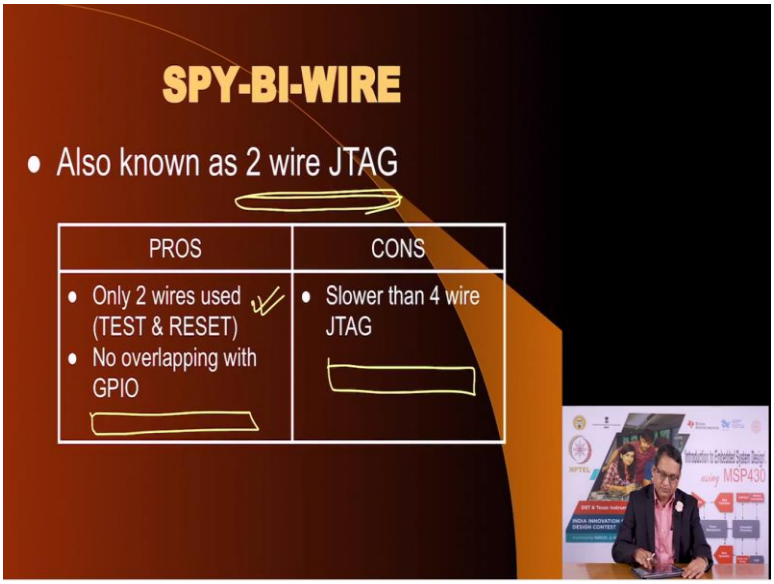
Now, if you choose to have access to MSP430 launch pad which is an official evaluation kit from MSP from Texas Instruments if you choose to do that, you would realize that on your evaluation kit, there are actually two big microcontrollers. And one of them is the target microcontroller that you are playing with the entry level MSP430 Launch pad does use MSP430 G2553 the same as we have on our on our kit.

But apart from that, you would see that there are a bunch of ICs and one of the ICs is actually the one which connects to the PC through a USB interface. And that IC involves connecting to the target microcontroller using this approach. It invokes the Spy-Bi-Wire protocol. And using these two wires that is test and reset it is able to download code into the memory of the target microcontroller.

And not only program it but you could debug your program by single stepping through the code, which means execute one line of program, stop there to see what is the result of that program by inspecting the registers or variables, and then going to the next instruction and so on and so forth. If you choose to have an external debugger, it would require more pins to be connected, and G2553 has access to those pins. But once you do that, you may not be able to use those pins for connecting to your peripherals.

And as I mentioned, the third method is the USB UART based Bootloader, which we are going to which I am going to explain in a little bit of more detail compared to what I discussed in what I discussed in an earlier lecture here.

(Refer Slide Time: 12:46)

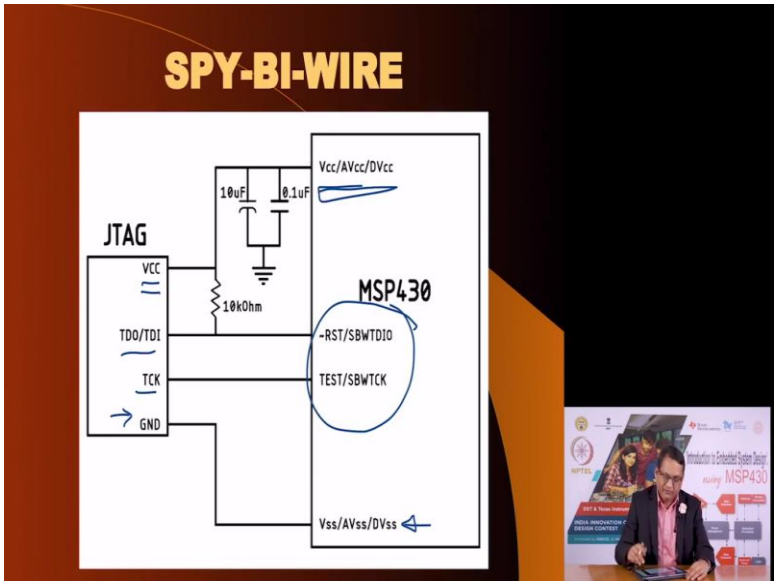


Now Spy-Bi-Wire is also called a 2 wire JTAG interface, because although it uses internally inside the MSP430 it invokes the J tag, but the external connections are just through two pins and therefore the advantages is that the interface is simple just involves two pins and there is no overlap with GPIO pins.

In the case of JTAG if you choose to use the proper JTAG interface on MSP430 microcontroller, that means that you would have to sacrifice 4 wires which are which share functions which share functionality with other features of the MSP430 microcontroller. And these four pins will be labeled TDI, TDO, TMS and TCK. But if you choose to use the Spy-Bi-Wire protocol, the advantage is that it is not going to come at a cost of using any GPIO pins because it uses two wires test and the reset.

And therefore, you are left with all your GPIO pins. The only disadvantage is because internally these two wires are as if being received by shift register which receives the clock and data on these two wires and make a full four wire protocol. So it will be slightly slower than the JTAG interface. But that's a small price to pay for the advantages that you get using this approach namely, just two wires and no GPIO pins sacrificed.

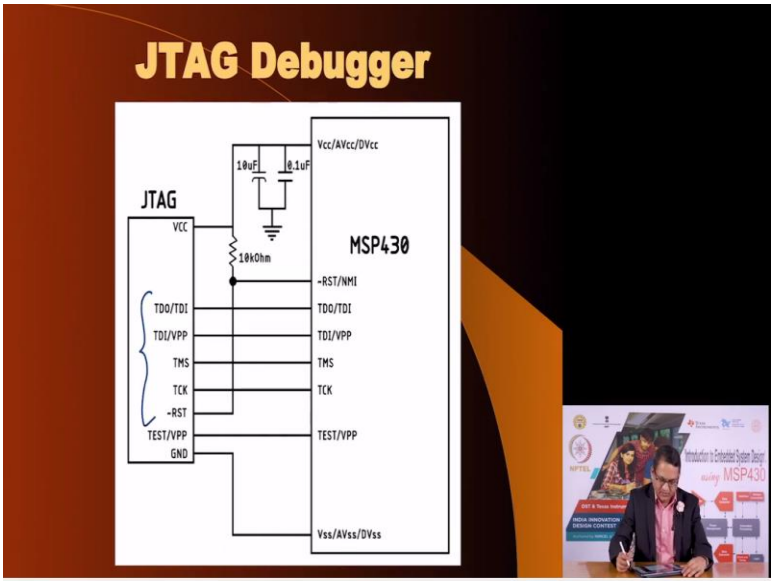
(Refer Slide Time: 14:20)



This is what the Spy-Bi-Wire protocol looks like that is you have to connect the supply voltage here as you see two available voltages and usually they may be available from the JTAG interface here. Then you have to use the reset and test pins of your microcontroller to the relevant pins of the JTAG interface from the Spy-Bi-Wire capable output, and of course connect the ground here.

And then using these two wires you can program and emulate the program inside the microcontroller after your downloaded it. So this is one option. And this is typically used on the MSP430 launch pad, which is an official microcontroller platform.

(Refer Slide Time: 15:12)



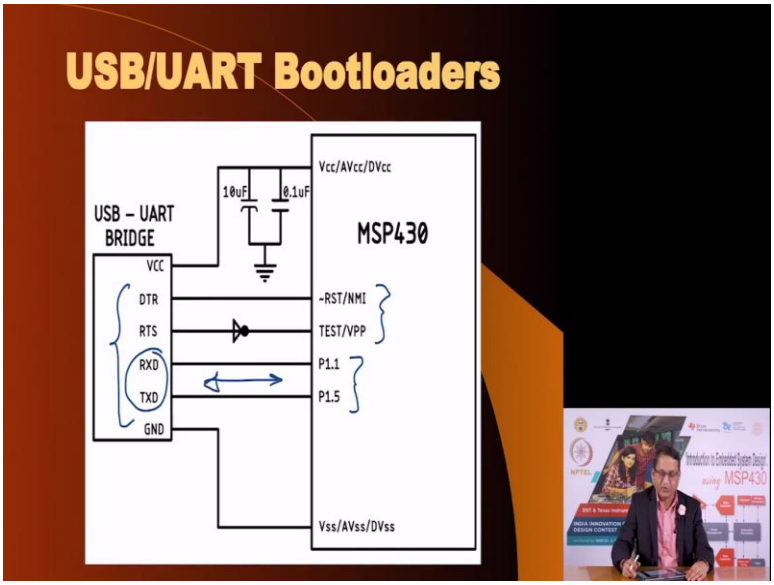
The other option is that you have a JTAG interface and there are many JTAG interfaces available. And then you would make connection with all the four wires that you see here, apart from the reset pin so that you can put the microcontroller in reset and then invoke the JTAG interface.

We do not recommend using it in this course of course in fact, we do not even recommend using the Spy-Bi-Wire but I mentioning it because this is if you have access to a launch pad you this is how it works on a launch pad. We are going to use this boot Bootstrap Loader, which Texas Instruments used to call Bootstrap Loader.

(Refer Slide Time: 15:50)

But now they just call it Boot Loader. It allows you any programmer to communicate with the MSP430 microcontroller and it is very good for prototyping and with this you are able to program the flash memory and also the RAM if you so wish.

(Refer Slide Time: 16:12)



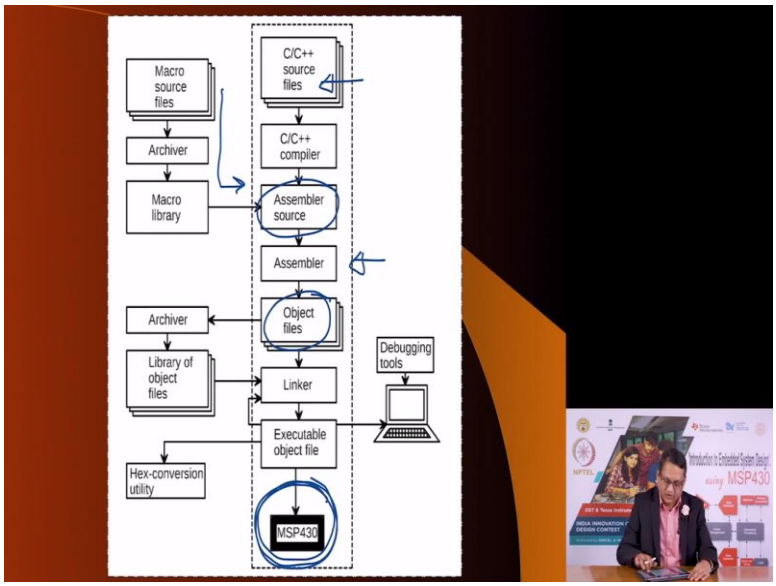
These are the connections that are required for the USB and UART based Bootloader that you need a mechanism you need a bridge as I mentioned earlier, a bridge which on one side connects to the USB and then generates the TXD and RXD communication signals. But on MSP430 there

are two pins designated as TXD and RXD for the purpose of Bootloader programming you do not connect to them you connect instead to these two pins.

And apart from that you also connect to the test and reset which the bridge invokes and asserts before proceeding with downloading the code through this, these two wires. And of course you have to connect to the ground and of course power the MSP430 microcontroller.

Now, we need to go through what is the program flow, how we are going to start with the editor, the IDE write our program, compile it or add third party libraries and so on. So I am going to go through that.

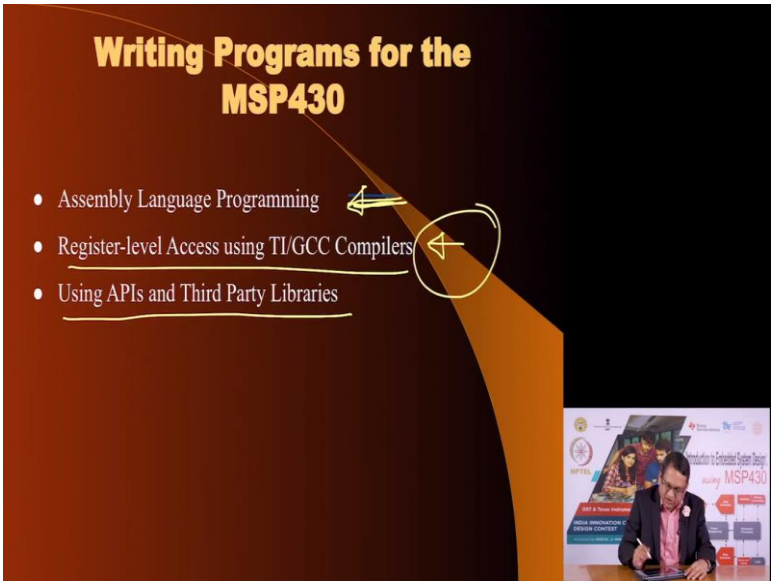
(Refer Slide Time: 17:24)



So you start with your C source file, you invoke the compiler options and the intermediate step in the compile process that it will give you the assembler source at that time, you could use third party libraries to integrate into this then this is assembled by the IDE and you get create object files. These are linked and you create an executable file which is nothing but the binary code and then you are ready to download it into the MSP430 target.

So, this is the flow of your code starting from what looks like as readable text which is nothing but C C++ commands. And eventually it is all compiled in various steps to be able to download into the MSP430 microcontroller.

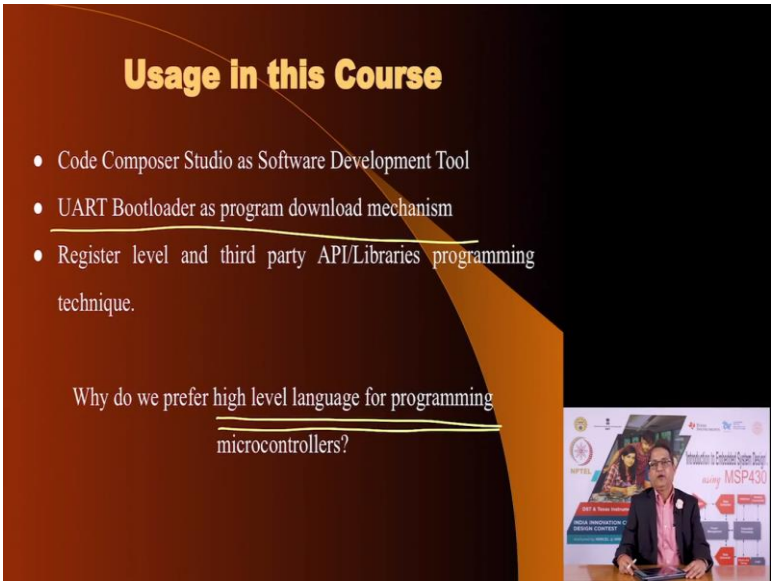
(Refer Slide Time: 18:12)



Now you can write your program on MSP430 for MSP430 irrespective of the, the platform, meaning the development environment, which could be CCS or GCC or it could be IAR and you could choose to program in pure assembly language if you so wish. We do not recommend that in this course because then your focus will shift from application development to assembly language programming.

You could also use register level access using the GCC or the TI compilers and we are going to take this approach. And the third option is you use APIs and third party libraries such as may be available from open sources and we are going to stick to this approach.

(Refer Slide Time: 19:00)



Now in this course, we are going to use the Code Composer Studio for our development, we are going to use the UART based Bootloader. And we are going to program using register level as well as third party API libraries involving all of them together. One question would arise that why do we concentrate on high level language programming, and that is because a high level language programming allows you to conceptualize and recreate complex create complex data structures, which you can easily manipulate in a high level language then it would be possible in the assembly language program.

Assembly language program might be more efficient in terms of the actual memory usage. But these days we are not short of program memory space and therefore, that is not really a big concern. Our more concern is, as I mentioned in one of the earlier lectures that time to market that we would like to start developing a project and quickly compile it converge, do all the hardware prototyping development and integrate the hardware and software aspects together.

So, time to market is an important consideration. And therefore, using a high level language to target your application development is a preferred method these days and we are going to use that. I hope this was beneficial to you. I will see you in a next lecture where we will deal with other aspects. Thank you