

**Introduction to Embedded System Design**  
**Professor Dhananjay V. Gadra**  
**Netaji Subhas University of Technology, New Delhi**  
**Lecture 21**

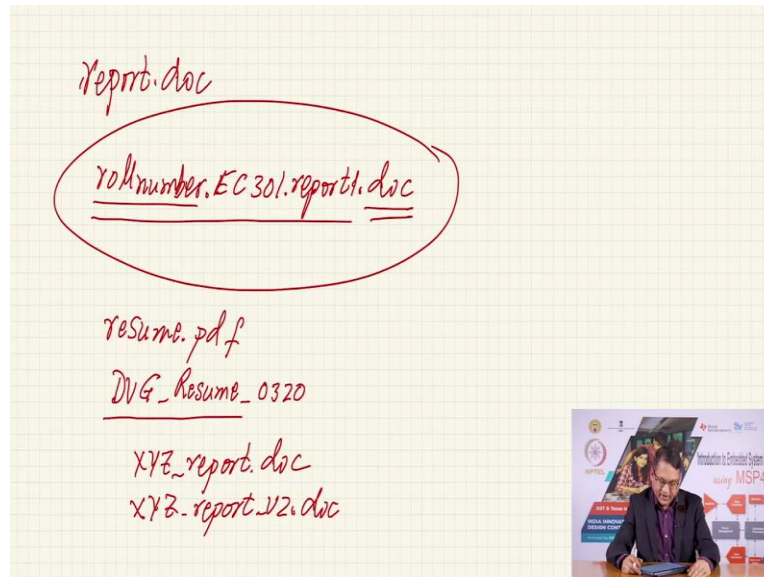
**GIT, CCS Installation and Embedded C**

Hello and welcome to a new session for this online course on Introduction to Embedded System design. I am your instructor, Dhananjay Gadre. This is a very important session in which we are going to go through certain steps. So, that appropriate software is installed on your desktop or laptop computer, whatever you maybe using. So, we are going to deal with two installations, one is a software called GIT, and the other is the code composer studio, which is the C compiler from Texas Instruments for MSP430.

In fact it supports many other macro controllers, but we are going to... restrict ourselves to discussion about the MSP430 micro controller. After the installation of these two packages, we are going to discuss a little about Embedded C programming. Of course, we assume that you have some experience, you have good experience in general C programming. So, we are not going to consider those aspects. We are going to consider specific aspects to the Embedded C part. Alright. Now GIT is a version control system.

Now, before I explain, what a version control system is, I would like to take this opportunity and discuss a few issues with you as a beginner student or as a student who has not yet entered the professional domain. I as an educator with some experience, I would like to share my experience with you. For example, suppose I. when I teach a class and I tell them, please send me a report about today's exercises that I covered in the class, most of the time unless I tell them to begin with, I would get 20, 30, 40 emails with a document enclosed, and often times the document is labeled or named report dot doc.

(Refer Slide Time: 2:18)



Now, as far as the student who has sent this is concerned, he is happy he has created a document and he has emailed it to me. But you please step into my shoes and consider that I have received 50 emails and all the 50 emails have pretty much 50 files. All of them are named report dot doc. If I download and save them, what is going to happen? It will make me crazy. It will be impossible for me to relate a given document with a person. And therefore, I always ensure that when I ask my students to send me reports for a given topic on email, I ask them to follow a certain protocol.

In a classroom, in a as a student you would have a roll number, and therefore often times my protocol is, you include this information in the file name. You say roll number, whatever your roll number is. The year in which we are doing, or the subject we are doing. Suppose we are doing EC301 and maybe if this is your first report, or if this is the first report that I have asked this class to send me, it could be report 1 dot doc or any other file format if it is a PDF file.

This way, the moment I get this, now roll number has to be replaced with your actual roll number, whatever it maybe. You do not want to receive 50 emails with the same roll number as a literal word. Here the roll number will be replaced by whatever the student's roll number is. Likewise, if tomorrow, now this is fine, this scheme works as long as you are part of a ecosystem where the roll numbers can be used to distinguish person A from B and C and so on and so forth.

Now, let me put you in a different perspective. Suppose tomorrow you are going to enter the job field, job market and you are going to send your resume to somebody. And that somebody has asked 20 other people to send their resumes. Now my question to you is, what would you send you, what would be the name of the file that you send to that person containing your resume. Well, if it is going to be resume dot PDF, well, you have recreated the situation which I explained earlier.

And therefore, what I normally do when I, when I go on speaking assignments to various colleges for giving a lecture, I have followed a format and I recommend to you that you follow a similar format if not the same. I always keep track of my changing activities and reflect them in my resume as I would say DVG, my initials, but that is because these are very uncommon initials. If you have names which end up having very common initials, please modify it. And then I say resume or cv. And then I also have a 4 digit number, for example, it could be 0320.

And here 0320 s going to tell me that I created it in third month of the year 2020. And this way, I am able to know, what is the latest resume that I have kept on my files. And the person who receives it can make an idea that this is a resume for a person whose initials seem to be DVG. You can follow a similar or a different format. But the important point is, there should be enough variety, enough information in the file name itself for somebody who is collecting tens of resumes to call one resume file apart from the other one.

Now, this is not the topic of our discussion here, but it is very closely related. Now imagine, so before that I hope you will give some heed to this idea that I have thrown at you. Now, coming back to the whole idea of version control system, imagine that you have a PC and that you are creating a, you are creating a report like your teacher might have asked you to create. Then you write a few pages of a Word document and then are happy with it, you save it, you go to sleep, next day you wake up and say, no no no, I think I can improve this report a little bit.

And so you open that file and you start editing it and you spend another few hours and suppose that report was xyz to reflect your initials, report dot doc. Alright. And then next day, you have modified it. You have edited large parts of it and you have replaced it with some new text. And few hours later you realized, oh god, the first version that you wrote last night was much better

than this one, now what to do. You have deleted parts of it and you have replaced it with new text, and you do not like that text.

How would you solve this? Well, if you are going to do manage this manually, you could say I, the night that you sleep happy with the report that you have created, the next day when you want to edit it, you open that file, but now first save it into a new file name. Well, at least some, keeping some part of the filename common, but an identifier which tells you that this is new version. So, what I could do is, I could say xyz underscore report. The next day, I will say underscore v2 dot doc.

And now, I have my file xyz report dot doc on my folder. I have created a copy of that and I renamed it as xyz report underscore v2 and I can edit it. And if I am happy with v2, I can retain v2. If not, I can always go back to the original xyz underscore report. And this way you can keep on creating newer and newer versions of your report.

Keeping the newer versions, if you are happy with it and reverting back to the older versions if you are not happy. And so, these stages while you start from the beginning if your file to some conclusion, conclusion of this whole exercise, you have created perhaps multiple versions of that file.

And this allows you to at least keep track of the changes that you have made. Now, of course, if you did it manually, it will be a nightmare to have so many files on your system. This is where an automatic version control system or creating documents under the supervision of a software which keeps track of these changes, comes into picture. Now, till now what we discussed was in the context of one person, you as an individual creating documents and you could extend this analogy to not just creating documents but also to create software of files.

You write a piece of code, you are happy with it and you want to make changes, you make changes which ends up making it worse than what it was earlier. So, when I say documents, it could be Word documents, or it could be Text documents, it could be code files, it could be anything else. This whole discussion applies to all of it. So, imagine that you were working on a project in which there were three or four files and it was being worked on by a few people, who were not geographically present at the same location.

That is they were distributed across the town or across the city, or maybe across the world, which is quite common these days. Now, imagine that all of you wanted to access files, make changes to those files and it is possible that the changes that you made, made the whole new version worse than what it was earlier. And so, you not only need to create, you not only need to take care of these files, the changes that you are making, through a automatic piece of software, but you also need a single source or single repository, single location where these files will be accessible to all the people who are working on it.

And in fact, the whole Cloud based storage of files maybe a Google doc is an example of that, but in this context we are going to use specific software created to facilitate keeping making newer and newer versions of the project that you are working on, whether that project is writing a book, writing a report, creating a schematic file, creating code software, no matter what, this version control system can take care of all of it.

Now, there are of course a lot of version control systems. Some of them are paid, many of them are paid, but there are some which are open source and free for you to use. And we have selected one such, which is quite popular and quite common, and that is called GIT. Now, what we want you to do is, we want you to become familiar with GIT and you can become familiar with GIT by beginning to use it. If you already know GIT, well you can skip this initial part of my, of this session, and head on straight to the part where we start discussing the code composer studio.

But if you are not, this is a right place to start. And GIT is a very complex software. It is, it offers you lot of facilities. It offers you facilities as a creator. It offers you facilities as a end user. And right now, the best way to start using it if you have not used it before is to start using it as a end user. And you as a participant, we assume that many of you or most of you have not heard of GIT, or version control softwares. And so, we are going to show you the use of GIT from the perspective of a beginner.

As you start using it, there are lot of resources which will teach you about various features of GIT, specially the features of GIT, when you want to use it as a creator and the material that you have created, you want it to share with others, or share with other members of the team, or share with common public, you could use the advance features. But right now, we hope that you are a beginner and so, you would like to, we would like you to install GIT, and using this, we would

like you to download all the code files and in fact schematic files and other things that we have shared on a repository that we will soon talk about, and it will allow you to download all of it on to the computer that you are using.

(Refer Slide Time: 13:16)

**Software link**

- <https://git-scm.com/downloads>
- Follow normal installing steps

**Initial configuration of user**

- `git config --global user.name "Username"`
- `git config --global user.email "email.id@domain.com"`

**Initial configuration of us**

- `git config --global user.name "Username"`
- `git config --global user.email "email.id@domain.com"`

*git config --global user.name "dugadre"*  
*git config --global user.email "dvg@gmail.com"*

Alright. So, to begin with, open a browser of your choice that you are using and type this link which is https double slash GIT dash scm dot com slash downloads. Once the software is download, downloaded, you can double click on it and follow the normal installation process. Once the software is installed, you open a terminal on your Windows PC. By the way all this

discussion is a Windows PC centric. That is the most common operating system for people to be using. And so we are going to discuss all of that in the context of Windows.

So, open a terminal and in that terminal, find a suitable, create a suitable folder into which you would want to show, save all the files related to this course. So, I leave it to you to create that folder. And using the terminal go to that folder, and then type this, type GIT config double dash global username and then username. So, for example, if I were to do this, I would on my command prompt on the terminal window, I would say GIT, I would type this, config space double dash global user dot name space, and then I will write the name that I want.

Suppose I want to do it as my name, so I could say DVGADRE, I close those and I press enter. After that, so this registers me with GIT with this username. And then I say on the command prompt again, I say GIT config, basically repeat most of it, double dash global user dot email double quotes, whatever is my email address.

So, if it could, if it was DVG at the rate Gmail dot com, I should type that, or if it is something else, I should type that. By the way, this is not my email address. So, you once you do this, it will basically register this software with this username and assign this email address to this user. Now what do you do?

(Refer Slide Time: 15:53)

**Create Repositories**

- git init ←
- or
- git clone <url> ←
  - For example: git clone [https://github.com/ticepd/EmbSysDesign\\_NPTEL\\_Course.git](https://github.com/ticepd/EmbSysDesign_NPTEL_Course.git)

---

**Handling changes**

- git add <file>
- git commit -m "elaborate changes message"
- git log
- git log --follow <file>

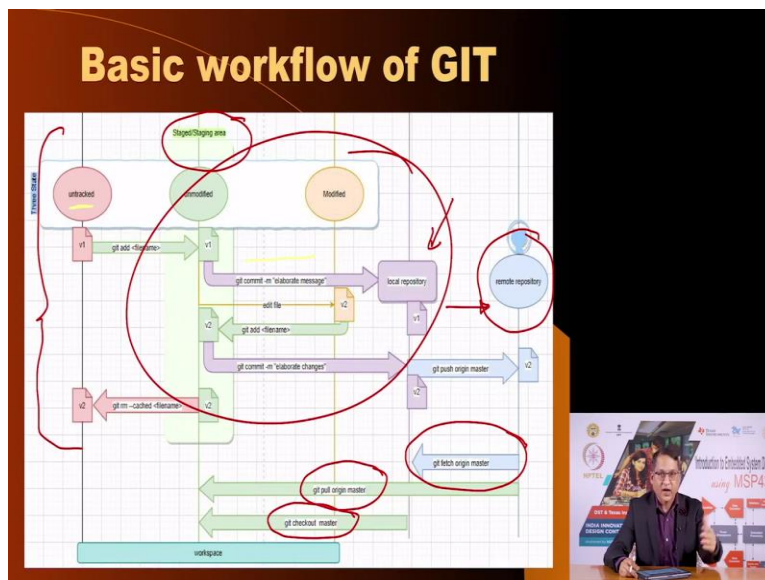
The slide also features a small video inset in the bottom right corner showing a man in a suit sitting at a desk with a laptop, with a presentation board in the background that includes logos for NPTEL and IIT Bombay, and text such as 'Introduction to Embedded System Design' and 'MSP430'.

Now again, in that folder that you have created, you could do one of two things. You could execute this command on the prompt. But this would you would do if you are somebody who is creating a repository, you are creating a set of files that you want to eventually share with the rest of the world.

Since you are not that, you are going to execute this second line. And what do you do, you type GIT clone, GIT clone and a URL address. So, which means basically you are going to write GIT clone followed by this entire address. Following the letters and characters exactly. If something is in capital, you must do that in capitals. And when you enter this, basically it is going to pull from a, a repository on the web internet called GIT hub. We have created this repository, it will create a local version of the files that are stored on that repository on to your local computer.

So, that you could access them even if the internet connection is not on. I will talk about more about this. Then if you are going to make changes to the files and you would want to share the changes with us, of course we need to agree to those changes and we need to agree to that, then you could follow these things. But we are not going to right now do that. We will talk about this in next slide.

(Refer Slide Time: 17:18)



Now, when you create a repository or you create a file, basically GIT looks at those at that file in various ways. When you create a file, it is still not attached to that software which is, which

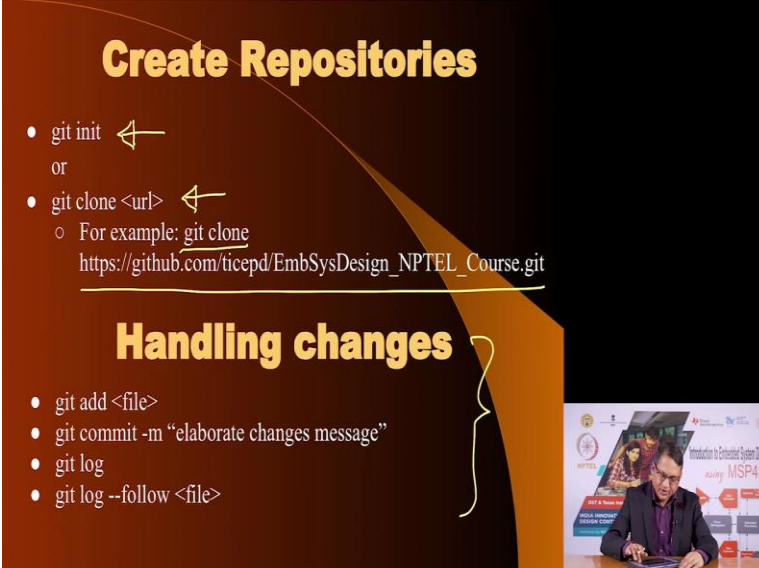


you just installed on your PC, that is GIT. Now, you can from that and that such a file is called untracked file. This is the untracked file. Now, and you obviously created in that folder, that I mentioned. And then, you can add that file into this next stage which is called the staging area. And from there, you commit it and, and now all this initial these three parts here, these are when you are doing it as an experienced user who is creating a repository, first of all that repository has to be created.

This is the repository on your local computer. And then you upload it to the internet global repository. So, when I say install GIT, that GIT is a local repository. Now, this local repository could be cloned as the word they use, could be replicated on a platform that is globally accessible.

Because your computer is not accessible to other users. If you as a creator wanted to share your work with others, you could move your, or you could copy your repository from your local computer into, onto a global GIT, or a global version control software website. And one of the website is GIT hub. There are other websites also. But we are going to use GIT on our local computer and GIT hub for the internet access. So, you can push all those files on to the internet repository on the internet, which in this case is GIT hub.

(Refer Slide Time: 19:10)



**Create Repositories**

- git init ←
- or
- git clone <url> ←
  - For example: git clone [https://github.com/ticepd/EmbSysDesign\\_NPTEL\\_Course.git](https://github.com/ticepd/EmbSysDesign_NPTEL_Course.git)

**Handling changes**

- git add <file>
- git commit -m "elaborate changes message"
- git log
- git log --follow <file>

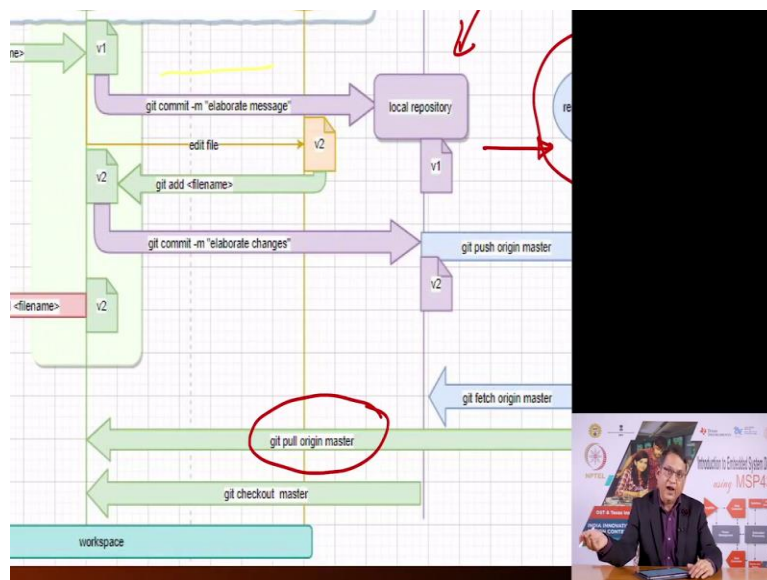
The slide also features a small video inset in the bottom right corner showing a man in a suit sitting at a desk with a laptop, with a presentation board behind him that includes the text 'Introduction to Embedded System Design using MSP430'.

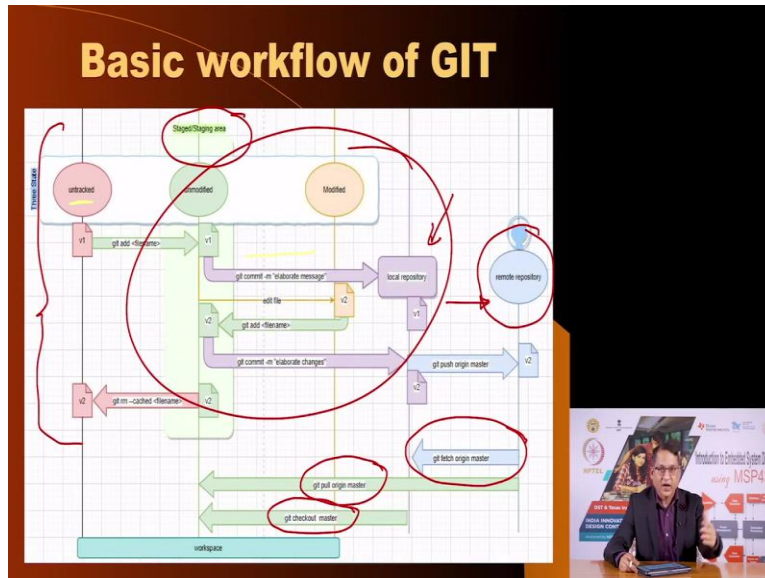
If you notice here, you install GIT, but you downloaded the repository that we have created on the GIT hub website. Likewise, when you, when you did this, when you cloned that repository that we have created on GIT hub, after that, you are set. All the files that are there on our repository will be replicated on your, on your computer, in the folder at which you did all this. And at this point you are ready to use them.

Now, imagine the following, that let us say you created this clone on your local computer and you are happy with it and then you went to sleep, maybe you stopped doing work for a week. And a week later, you wanted to resume. Well, you need not worry, you could open those files in the way that we will talk about soon.

But what if we here make changes to those files. If we make changes to those files, you would not know them, you would not get the chased version of the files, unless you did something. And what do you need to do, you need to go to that command prompt in the area that you, in the folder that you created and you need to pull those files again. Now, in case those files have not changed, nothing will change. In case we have made changes to those files, they would be reflected, you would get a newer version and the local repository on your computer will be updated with the changes that we may have made, we might have made.

(Refer Slide Time: 20:53)






And therefore, you need to learn to use two commands for that, usually you would run this command. You would say GIT, exactly like this, on the command prompt, you just type GIT pull origin master. The GIT software on your PC knows what you are talking about, what is the remote host, from where to get it and update which files. In case you do not want to and that will bring into the, the so called staging area, at which point you can use them in the way that you want.


You could also, of course when you do this, the local repository is also updated. Or you could use GIT fetch origin master, you can type this in the command prompt. It will only bring it into local repository. It is not yet available, these newer changed files may not be visible, will not be visible to you in the work area. And then you need to execute another command called GIT checkout master, for the changed versions which have now come to your local repository, but not yet available or visible into the designated folders.

So, we suggest that each time you are working on these files, you just open your computer, make sure that the internet is working and then go to the folder, go to the, open the terminal in the folder that you have stored all this and just run one command which is get GIT pull origin master. When you type it, any changes that may have happened would be reflected on the local repository and into the work place where these files are stored.

(Refer Slide Time: 22:27)



**Synchronize changes**

- `git fetch` 
  - Downloads all history from the remote tracking branches.
- `git pull`
  - Updates your current local working branch with all new commits from the corresponding remote branch on GitHub.

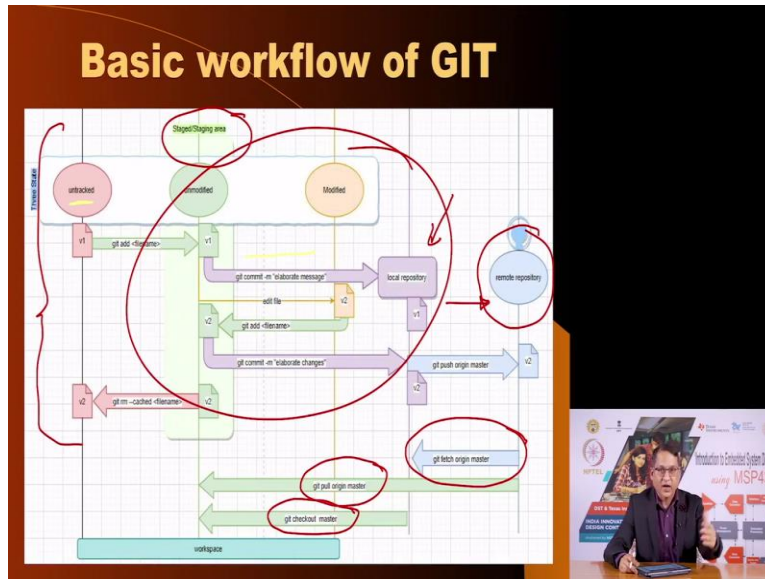
The slide is part of a video lecture. In the bottom right corner, a small inset shows a man in a suit sitting at a desk with a laptop, speaking. Behind him are several posters, including one for 'Introduction to Embedded System Design using MSP430' and another for 'MSP430'.

And so you say GIT fetch. It allows you to fetch these files and you can do, but we recommend that you do GIT pull, which will not only fetch it, but also store and make the files visible in those folders where the original files were stored. So, this is the way to ensure that any changes that have happened on the GIT global repository, those changes are reflected in your local repository. So, let me revise some of these terms. GIT is a version control system. There is a local version of it running on your computer when you install GIT.

There is a global repository and there are many such global repositories and GIT can talk to any such global repositories, but we have chosen to connect GIT to a global repository called GIT hub. GIT hub is where we upload the files that we are creating for you. And if you want to synchronize the files that you have already downloaded on your local repository, you need to execute this command. So, on your terminal you just type GIT pull with this word GIT pull original master, it will bring the latest version of the file from the global repository, that is GIT hub, in from that file, that link that we shared.

You do not have to provide that link, it knows. It will simply pull those files from that repository and update your local repository and then you are free to use them in any which way you want. You can make changes to it also, does not matter. Of course, those changes will not go and change the files on the global repository.

(Refer Slide Time: 24:14)



For that you need to learn more about how to use GIT and essentially that part is covered here. So, you could use this as a reference, learn more about GIT. There are lot of sources of learning material related to GIT on the internet. We recommend that you read that if you want to be a contributor, or later on if you want to create your own repositories and share it with the world, or create it, or have a team in which multiple people need to work on such shared files. So, once you are done this, you are ready to go to the next step, which is to install code composer studio.

(Refer Slide Time: 24:50)

## Clone remote repository for this series

```
git clone https://github.com/ticepd/EmbSysDesign_NPTEL_Course.git
```

## Download files directly

```
https://github.com/ticepd/EmbSysDesign_NPTEL_Course/archive/master.zip
```

The slide displays the command to clone a remote repository and a direct download link. The slide also features a video thumbnail of a presenter.

And just to recap, you have created a clone using this command. If you just want to download the files once and for all, without the possibility of synchronizing with the changes, you could download just the files using this command, type this into your browser and you get the zip file which has all the folders and the sub-folders and the files inside those and so on.

Of course I hope that you are going to be worried about the synchronization and all that, so you would use this command. Well, now you are ready, your GIT is installed on your computer, you have downloaded the global repository and you are now ready to jump into the next part of this, which is to install the code composer studio.

(Refer Slide Time: 25:37)

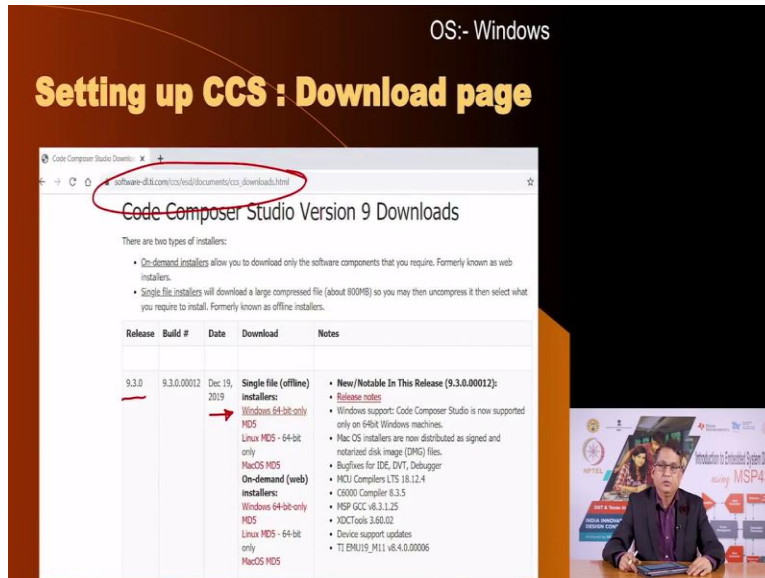
software-dl.ti.com/ccs/led/documents/ccs\_downloads.html

## Code Composer Studio Version 9 Downloads

There are two types of installers:

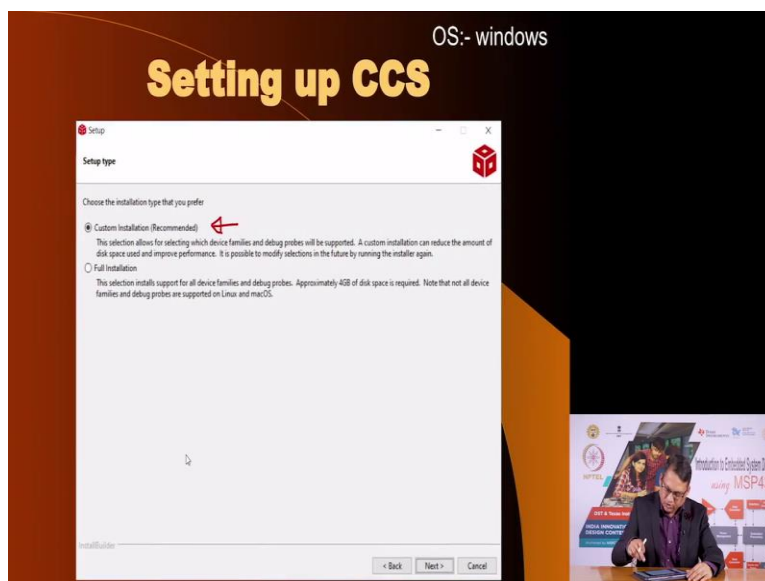
- **On-demand installers** allow you to download only the software components that you require. Formerly known as web installers.
- **Single file installers** will download a large compressed file (about 800MB) so you may then uncompress it then select what you require to install. Formerly known as offline installers.

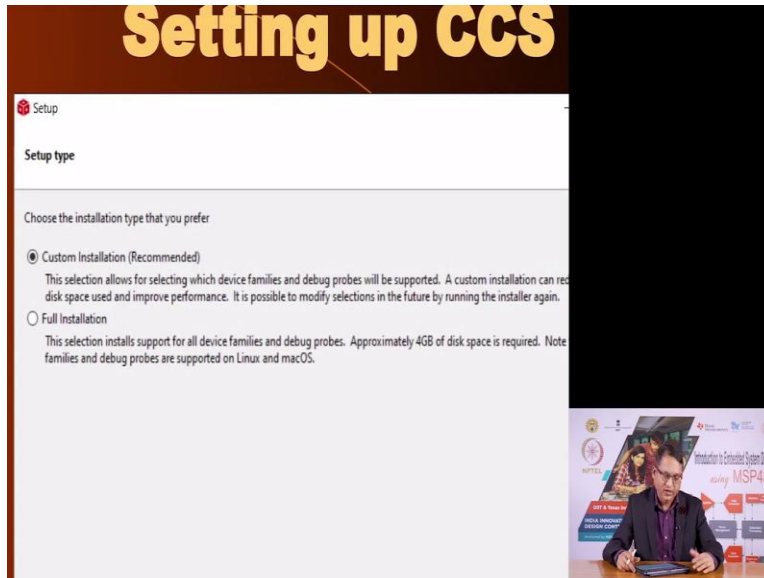
Release	Build #	Date	Download	Notes
9.3.0	9.3.0.00012	Dec 19, 2019	<b>Single file (offline) installers:</b> Windows 64-bit only MDS Linux MDS - 64-bit only Mac OS MDS <b>On-demand (web) installers:</b> Windows 64-bit only MDS Linux MDS - 64-bit only Mac OS MDS	<b>New/Notable In This Release (9.3.0.00012):</b> <ul style="list-style-type: none"><li>• <a href="#">Release notes</a></li><li>• Windows support: Code Composer Studio is now supported only on 64-bit Windows machines.</li><li>• Mac OS installers are now distributed as signed and notarized disk image (DMG) files.</li><li>• Bugfixes for IDE, DVT, Debugger</li><li>• MCU Compilers LTS 18.12.4</li><li>• C6000 Compiler 8.3.5</li><li>• MSP GCC v8.3.1.25</li><li>• XDCtools 3.60.02</li><li>• Device support updates</li><li>• TI EMU19_M11 v8.4.0.00006</li></ul>



For that, you go to this website, or you just search code composer studio, TI code composer studio, you would get the link. Go to this or type this into your browser and when you see, you will see this. There is lot of options, but the top version which is released 9.3.0 is what we want you to download. And we want you to download the Windows 64 bit version. There are no other versions available anyway, so you click on it and follow the normal software download and installation process.

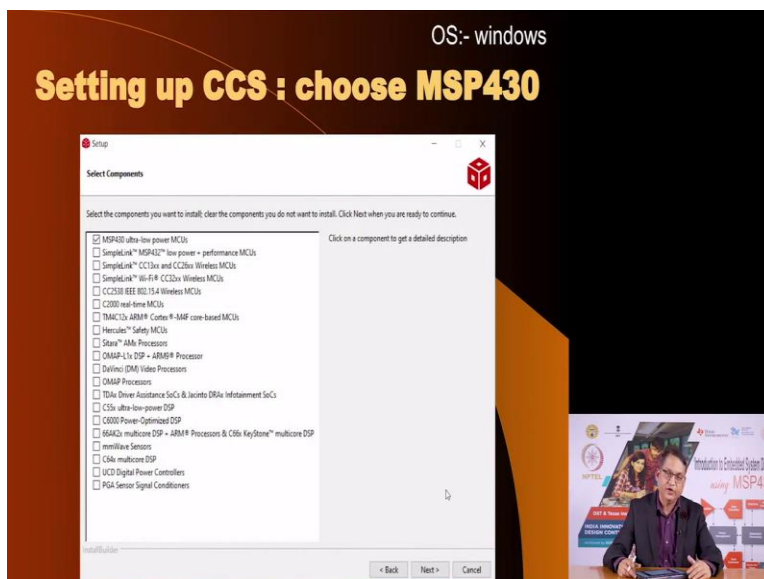
(Refer Slide Time: 26:05)



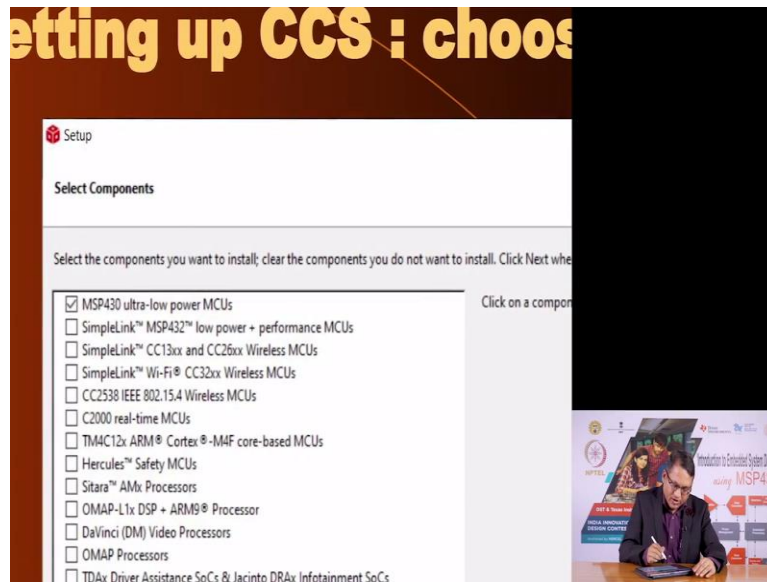


When you click on the downloaded part, it allows you to setup, it offers you two options. Either you can go with the custom installation or you can go with the full installation. As I mentioned code composer studio is C compiler, C, C plus compiler from Texas Instruments for a lot of their micro controllers and processors. But since in this course we are only going to be talking about MSP430, there is no need to for full installation. If you did a, if you chose the full installation, it will install compilation ideas for all the processor that TI offers. We are only dealing with the custom part that is one part out of so many processors. So, we recommend that you use the custom installation.

(Refer Slide Time: 26:50)

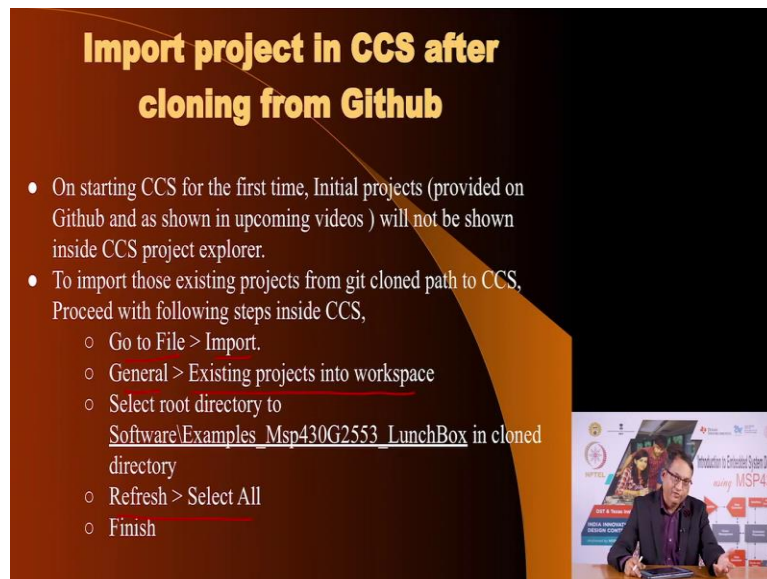






Then you go next, and in the next, it offers you all these options which you can check out. And since as I mentioned, we only want to deal with MSP430, check out the first option which is MSP430 Ultra low power NCOs. And then select next.

(Refer Slide Time: 27:08)



And now you are ready, your software is going to be installed shortly. And once it is installed, we want you to be able to write code files and compile them and download them and so on so forth. Of course we do not want you to write code files from scratch. In the beginning we would want you to use the files that you have just downloaded. Remember in the previous segment of

this lecture we talked about creating downloading a repository of code files and of course many other things on to your local computer. So, all you need to tell CCS that I want to edit existing files, you want to point where those files are located and so you need to import those files.

So, to import existing files, you go to file import and in the tab it will show general and then you say that you want existing projects into workspace and you need to point it to the right directory which would be in that folder that you created. And once you do that, you refresh select all finish. Now, you are good to go. You have all the your code composer studio would know all the code folders that we have created for you, they are already installed on your computer, but now code computer studio knows where they are and that you want to work on them.

In case, you want to create a new project, meaning apart from all the code files in various folders that we have shared with you, in case you wanted to create your own new project, you are of course welcome to do it.

(Refer Slide Time: 28:45)

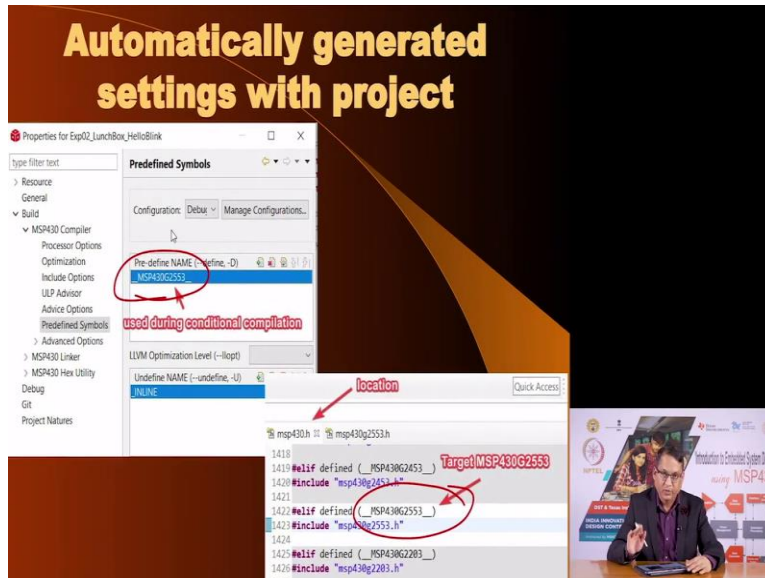
**Setting up a New Project for MSP430G2553**

Navigate using following steps:-

- File > New > CCS project

The screenshot shows the 'New CCS Project' dialog box with the following details:

- Target: MSP430G2553
- Connection: MSP430G2553 (Default)
- Project name: [Project name]
- Use default location: [Use default location]
- Compiler version: MSP430G2553
- Project templates and examples: MSP430G2553



And for that, you use in your CCS go to file, select new and you say you want to do a new CCS project. There you need to mention that your target is this. This is the MSP430 G2553 and that the compiler version is this. And then when you say finish, it will create, it will automatically install some information for you into specified areas.

And one of the important things it will store is that it you have chosen MSP430 G2553. So, in the predefined symbols, it is going to use this file. And in the header file, it is going to use this version. This is just to illustrate what is happening, You, do not need to be worried about it. Once you have selected this option while installation, all the files that you create, all the code that you write in these files will compile for the MSP430 G2553 micro controller.

Now, once you have created a file, or you have opened a existing file, you would want to compile it. Now, of course compilation will convert the C file into assembly and from the assembly into a machine code file. But what do you do with that machine code file. Your idea is, your intent is that this machine code, that is the object code, the binary code should be transferred into the memory of the MSP430 micro controller. And for that, as you know, you would connect your MSP430 lunch box.

But just the compilation process is not going to transfer the code file into the MSP430 target. I am using the word MSP430 target because it could be MSP430 lunchbox, or you could use any

other MSP430 evaluation kit. Right? And so we need to still tell MS, our CCS compiler that once the code compilation is completed, what is to be done with the object code.

What we want to tell it is that we want to invoke the boot strap and we can do that in the CCS. So, that I do not need to open ten windows and in each window I have to tell one part and the second window I tell the second part, the third part. The code composer studio is IDE, Integrated Development Environment, which means it allows me to edit, it allows me to make changes, it allows me to save files, it allows me to open files, it allows me to compile them, it allows me to look at the compiled versions and then eventually it allows me to download the software right into the target micro controller.

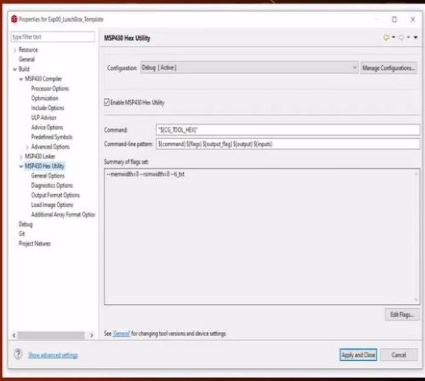
And so for that, we have to tell what is our way for transferring the object code from the compiled, from the CCS compiler into which way into the micro controller memory. And since we are going to use the boot strap loader, we need to instruct the, the CCS that this is our preferred method.

(Refer Slide Time: 31:39)


OS:- windows

## BSL settings in CCS

Step 1



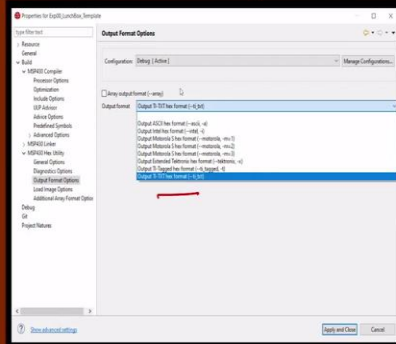
• Project Properties > Build > MSP430 Hex Utility > Enable MSP430 Hex Utility



OS:- windows

## BSL settings in CCS

Step 2



- Project Properties > Build > MSP430 Hex Utility > Output Format Options > Output TI-TXT hex format

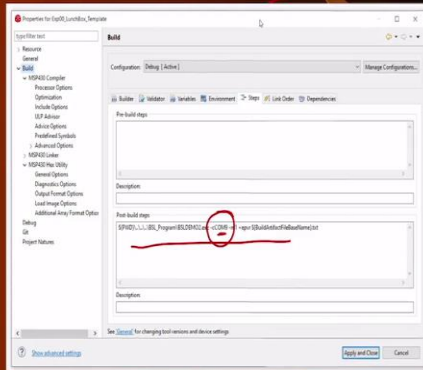


OS:- windows

## BSL settings in CCS

Step 3

- Project Properties > Build > Steps tab > Post-build steps
- `$(PWD)\..\..\BSL_Program\BSLDEMO2.exe -c COM9 -m1 +epvr`  
`$(BuildArtifactFileBaseName).txt`



And for that, you go to project properties, there you would see a tab build. In that you select MSP430 hex utilities and you say enable MSP430 hex utility. Then you press apply and close. And then you do the second step, where you again go to project properties, you say build MSP430 hex utility. You are specifying the output format options and you select, here as you see, output TI dash TXE hex format.

There are many ways in which the hex file can be saved. This is the preferred version which is compatible with the boot strap loader, so we want you to select this. Once you are done, then you go to step 3. And you, it basically shows you that once the build is completed that is when I say

the word build, here it means the entire process of compilation and so on and so forth. What to do with it? It is going to create a filename like this. and remember, this part is very critical.

This is basically telling the computer and the code composer studio that the compiled file has to be downloaded, has to be sent through the USB port which is seen as by the computer, which is seen as com 9 in this case. Right? But of course, in your comp, on your computer the USB port which you connect the lunchbox may appear to be a different number. So, what you need to do is, go into the device properties and find out that the port, when you connect your lunchbox, it is reflected as what com number.

And the way to do that is, if you have already connected the lunchbox, disconnect it. Connect it again and you see a number pop up into the device manager. And that is the number. And you should change that, that, you should write that number here. And this is when you are going to be installing a new file. If you are going to open the files that we have created and which are available to you on the repository, you might find this number and this number may be different from the one reported by your device manager.

So, you need to go back into this BSL step 3 and edit this part to reflect that com port number that you see now. And one idea is that if you have chosen a particular USB port to connect your lunchbox. If you continue to use that port over the period of this course, you may not have to worry about having to change this over and over. Five days later also if you open your computer and connect it to the same port, the port number would not change.

But just in case, your code does not seem to work on the lunch box, one of the reasons could be that this port number has changed. So, it is not a bad idea to check with the device manager to find out, is this port number has changed, since the time that you set this in this BSL settings. If it has changed, please make appropriate changes here. So, it is very important that you find out the port address to which you are going to connect your lunchbox and if it has and that address must reflect into the BSL setting in this step 3. Once you are done, you are good to go.

(Refer Slide Time: 35:12)

**MSP430 C/C++ compiler**

The TI compiler accepts C and C++ code conforming to the International Organization for Standardization (ISO) standards for these languages. The compiler supports both the **1989 and 1999 versions of the C language** and the 2014 version of the C++ language.

The slide features a dark brown background with a curved orange shape on the left. A small inset image in the bottom right shows a man in a dark suit sitting at a desk with a laptop, with a presentation screen behind him displaying logos and text including 'MSP430' and 'Introduction to Embedded System Design'.

Now, let us talk about the MSP430 compiler. It is a compiler which accepts C, C plus plus commands. And it is god for 2014 version of the C plus plus language.

(Refer Slide Time: 35:27)

Level :- For beginner

## Keywords in C

The MSP430 C/C++ compiler supports all of the standard C89 keywords, including const, volatile, and register. It also supports all of the standard C99 keywords, including inline and restrict. It also supports TI extension keywords `__interrupt`, and `__asm`. Some keywords are not available in strict ANSI mode.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

The slide features a dark brown background with a curved orange shape on the left. A small inset image in the bottom right shows a man in a dark suit sitting at a desk with a laptop, with a presentation screen behind him displaying logos and text including 'MSP430' and 'Introduction to Embedded System Design'.

Now, we are not going go in this, C of course has keywords which you cannot use in as names of your subroutines and things like that. And here is a list of those keywords. Please avoid these. Please, please understand these, please familiarize yourself with these keywords to ensure that you do not use them when you are writing a program.

(Refer Slide Time: 35:50)

**Main function**

- Entry point
- Its compulsory.
- Good practice is not to exit it. (use while loop)

*ABC: jmp ABC*

**General main() code flow**

- Configure WDT ✓
- Set up the clock ←
- Configure ports and peripherals ←
- Enable Interrupts ←

*}*

The slide also features a small inset image of a man in a suit sitting at a desk with a laptop, with a presentation slide titled 'Introduction to Embedded System Design using MSP430' visible behind him.

Now, what does a C program have? At the very minimum it has a main loop. That main loop indicates the entry point into the micro controller or the micro processor, it is compulsory. And it will do a bunch of things as, as reflected by the code that you write. But once all those commands, all those instructions are executed, where, where does the main loop take you? Well, actually it does not go anywhere. The micro, as long as the micro controller is powered, it has clock, and it has memory, it will continue to execute some program.

You may think that the instructions that you have written are over, the micro processor is now sleeping, it is not doing anything. But that is not the case. It actually has infinite loop where it is jumping to itself. So, it could be something like this, let us say the label is ABC. And it could say jump ABC. So, it is like, the equivalent assembly command would be jump to itself. So, it will be waiting, it is doing something. For you it may not be doing anything, that is visible, but it is actually doing something.

And so you can explicitly have a infinite while loop, after your code instructions are as you understand is over. Or if you do not, the C compiler will actually be executing some instructions. Now, what is inside a main, the main loop. Well, what we do in the context of MSP430 is that we will cover this, this is a watch dog timer. It is a very important feature of any micro controller. And you need to configure it. So, you can choose to have it operational or you can choose to disable it, but you must deal with it.



Then a micro controller and specially MSP430 has lots of options of the various clock signals. So, you need to set that up, if you choose to. If you do not set it up, the MSP430 works with some default clock, which is usually born out of the DCO and the frequency is 1 point 1 mega hertz and I am talking in the context of MSP430G2553 micro controller. Then you need to do something to configure the ports and peripherals that is you may have output ports and you want to say, you have ports and you say I want these ports to be output, or some bits of those ports you want to be input.

Or some timer you want to work at certain frequency or some ADC you want to work at certain rate, you do these initializations. After this, in case your code requires interrupts, you enable them. And then comes the actual meet of the program which is basically you are doing something repeatedly.

(Refer Slide Time: 38:42)

## Structure of Embedded C code

```
1#include <Library/LunchboxCommon.h>
2#include <msp430.h>
3#include <stdio.h>
4#include <string.h>
5#include <inttypes.h>
6/*
7 * An example of multi-line comments
8 * @brief entry point for the code
9 */
10int main(void)
11{ //! Stop Watchdog (Not recommended for code in production
  and devices working in field)
12  WDTCTL = WDTPW | WDTHOLD;
13
14  initialise_SerialPrint_on_lunchbox(); // a function
15  int x = 0;
16  while (1)
17  {
18    x++;
19    printf("Hello world %d \r\n", x);
20
21    int i;
22    for (i = 0; i < 20000; i++);
23  }
24}
25}
```

As an example, let me show you a structure of a Embedded C program. As you see, initially we have all these input files which have the prototypes of the function that you are going to invoke and many other things. And here is the main loop. Now, why we are doing this is, not only we want to compare it with the description of the various elements in the main loop, but also something interesting.

So here this part, we have disabled watch dog timer and you would understand what we are doing here once we cover the watch dog timer later on. Then we have called a function, which is initialized serial print on the lunchbox. Now why we, why we are talking about it is, you see, when you learn C programming on a compiler, on your PC, you had a great mechanism to see the result of your program, and that was the display.

Unfortunately, on the micro controller, there is no dedicated display. There is a display very rudimentary display, but that display can show only binary information and I am talking about of an LED, that LED is on, it has done something, if the LED is off, maybe it has done something else. But there is no way to print verbose information, or it, it appears to be like that. Well, if you have used the PC or laptop desktop to download code into your micro controller, you could actually in principle, use the display of your development post, that is your laptop or desktop to display information.

For that of course you need, the micro controller needs to do a lot of things, it should be capable of doing serial communication and connect to the PC. But it does, that is how you are able to download the program from the C compiler into the memory of the micro controller. And so you could use the same channel through you which you programmed the code from the desktop into the memory of the micro controller. You could use the same channel and use it after the code is working to print some information on your desktop computer.

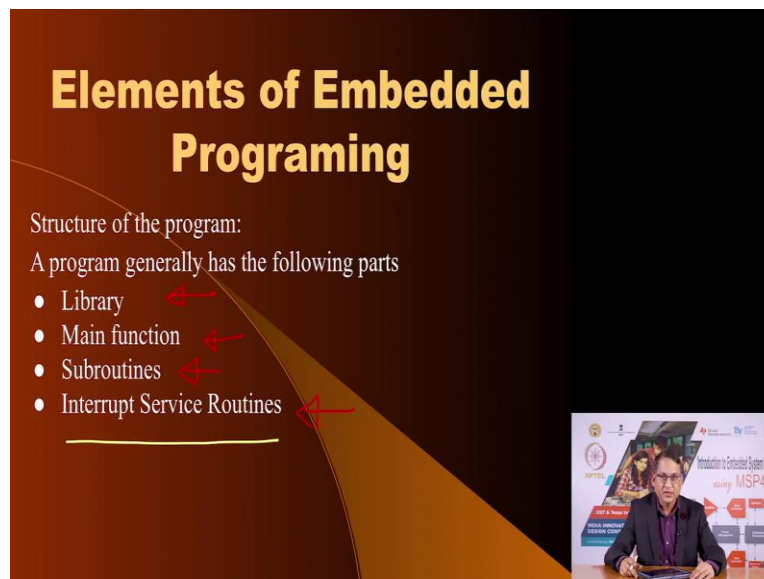
Of course, this would be useful only when you are building the code, when you are developing the code, this may, this info, this mechanism may not be available to you when the code is dully working and is installed in some application. But while you are developing the application, this is a great resource. And for that, all you need to do is include this file into any program that you write and then you could, as if you have the printer function when you did your first C program, you did Hello world right, and used this printer function.

It is like being able to use the printer function except it is not going to print it on the micro controller display, because your micro controller display does not have, your micro controller does not have a display like that. It is going to use the display of the PC and display information, and in this case, what it is going to do here in this case when you compile this code and

download it into the memory of the micro controller, it will print Hello world and followed by a number, and that number is going to continuously increment going from 1, 2, 3, 4 and so on.

And this part of the loop is simply just delaying it, so that you see that line for sometime and then that line is updated, it is pushed up and a new line prints. It says Hello world 2, Hello world 3 and so on so forth. And so this is to illustrate that we have some mechanism to debug the program, or use the display of the PC as a means of displaying verbose information, or more critical information, maybe values of some variables or some count that you want to display. If this would be, this part of the code function you would have to replicate in the code that you want to, that wants to use this facility. And so we are going to show you how to use it very soon.

(Refer Slide Time: 42:50)



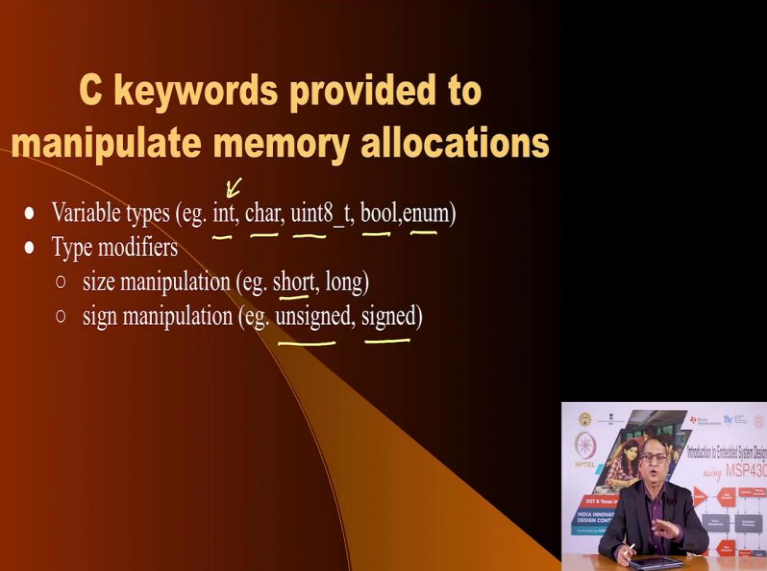
Now the, what are the elements of Embedded C programming. Well, I mentioned earlier, it has a library definition, it has a main function, of course only very simple programs would have only one main function and you can have only one main function. As you become experienced programmers, you would relegate many of the programming tasks to subroutines.

And so your program may have one or more subroutines. And if you are an Embedded programmer, that is what would distinguish you from conventional programmers is that you are going to include something called interrupt service routines. They are also subroutines except they will be invoked once when some event happens.

That event could be, you press a button, or the event could be you receive a value from the serial port, or the event could be that a internal timer reaches some value and so on so forth, which we are going to cover subsequent, in subsequent lectures when we talk about these peripheral features. But you are going to be adding interrupt service subroutines into the, the C program.

Now, what how you manipulate data in a program is very important. Because at the end of the day, the program is doing nothing but processing information which is data, you manipulate that data and you store that data back in the memory or you send that data into an output port and so on so forth. So, we need to worry about understand what are the various types of variables that are available to you.

(Refer Slide Time: 44:24)



**C keywords provided to manipulate memory allocations**

- Variable types (eg. int, char, uint8\_t, bool, enum)
- Type modifiers
  - size manipulation (eg. short, long)
  - sign manipulation (eg. unsigned, signed)


The slide features a dark background with a large orange and brown curved shape on the left. A small inset video shows a presenter in a suit speaking at a desk with a laptop. Behind him are several logos, including Texas Instruments and MSP430.

Broadly it is an integer variable, character variable. It is also, you can also look at it as unsigned integer of a size 8. Integer is usually 16, but you can over write that by saying no, I want to use an unsigned integer with, with 8 bits. You can have bullion type or you can have a numerated type of variables. Then you can modify the size by using, you know, before these variable you say short int, or you could say a long int that would modify the size of the int. And you could also use these variables which could be defined as unsigned or signed, it basically reduces the available numbers but allows you to have plus and minus numbers also.

(Refer Slide Time: 45:12)

## C keywords provided to manipulate memory allocations

- Storage classes
  - Manipulate lifetime and scope
  - Examples
    - ✓ ■ auto
      - It allocates local variable in stack memory
    - ✓ ■ static
      - Data will persist till program execution
    - ✓ ■ extern
      - Allow access outside current scope/file
    - ✓ ■ register
      - Allocates memory in cpu register. It's not so common, compiler does it better




Apart from that, the CQ words that are required to manipulate memory locations are auto in which variable is allocated in this stack memory, or you could have static in which the information will remain in a memory location. You can use an external definition in which you want to access certain memory locations which are outside the scope of the current program. Or you could use internal registers for storing information.

(Refer Slide Time: 45:41)

## Variables qualifier

- ✓
  - **const** : Declaring a variable as a constant means that its value cannot be modified by the program.
  - **volatile**: By declaring a variable as volatile, the compiler gets to know that the value of the variable is susceptible to frequent changes (with no direct action of the program) and hence the compiler does not keep a copy of the variable in a register (like cache). This prevents the program to misinterpret the value of this variable.
  - As a thumb rule: variables associated with input ports should be declared as volatile.

```
int my_value;  
volatile int my_value;
```



Also, the variables that you have defined, you could also qualify the way they operate, the way they are managed by the compiler by saying that the information is constant of course, then it is not a variable, but you are saying here is a number that I do not want to change. Or here is a number which is stored in some memory location. The micro controller often keeps a, could keep a copy of that memory location into a register. And so, whenever it is needed, instead of bringing it from the memory location or a port, it could actually give you a local copy.


And sometimes you do not want that local copy. You want the compiler to fetch it from the source, from right from the horse's mouth. And you have to tell the micro controller the C compiler that please bring the value of that variable right at the source. And one way to do that is to add this key word in front of the character or integer variable that you are declaring. For example, I could say `int my underscore value`. This would be a normal variable.

But if I said `volatile int my value underscore`. Well, you would still create a variable called `my value`. But you are saying that, you know, the information to be fetched from this variable has to be brought from that location where it is created. A copy if it is being maintained should not be used to provide you whenever you need it. And so usually we use this keyword when we are looking at exchanging information from an interrupt subroutine with the main program or from the main program into the interrupt subroutine. And of course you would get to know it when you see code examples that we discuss.

(Refer Slide Time: 47:35)

### Data types in C

Type	Size	Alignment	Representation	Range	
				Minimum	Maximum
signed char	8 bits	8	Binary	-128	127
char	8 bits	8	ASCII	0 or -128 <sup>(1)</sup>	255 or 127 <sup>(1)</sup>
unsigned char	8 bits	8	Binary	0	255
bool (C99)	8 bits	8	Binary	0 (false)	1 (true)
_Bool (C99)	8 bits	8	Binary	0 (false)	1 (true)
bool (C++)	8 bits	8	Binary	0 (false)	1 (true)
short, signed short	16 bits	16	2s complement	-32 768	32 767
unsigned short	16 bits	16	Binary	0	65 535
int, signed int	16 bits	16	2s complement	-32 768	32 767
unsigned int	16 bits	16	Binary	0	65 535
long, signed long	32 bits	16	2s complement	-2 147 483 648	2 147 483 647
unsigned long	32 bits	16	Binary	0	4 294 967 295
long long, signed long long	64 bits	16	2s complement	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
unsigned long long	64 bits	16	Binary	0	18 446 744 073 709 551 615
enum	varies <sup>(2)</sup>	16	2s complement	varies	varies
float	32 bits	16	IEEE 32-bit	1.175 494e-38 <sup>(3)</sup>	3.40 282 346e+38
double	64 bits	16	IEEE 64-bit	2.22 507 385e-308 <sup>(3)</sup>	1.79 769 313e+308
long double	64 bits	16	IEEE 64-bit	2.22 507 385e-308 <sup>(3)</sup>	1.79 769 313e+308
function and data pointers	varies (see	16			



These are the data types that are available in C. I do not need to repeat this as it is very similar to what you may have already studied when you learnt C programming.


(Refer Slide Time: 47:43)

### uintXX\_t Data types in C

These data types (Example → `uint8_t`, `uint16_t`) are included in the header file `"stdint.h"`.

It's very useful when you do computations on bits, or on specific range of values (in cryptography, or image processing for example) because you don't have to "detect" the size of a long, or an unsigned int, you just "use" what you need. If you want 8 unsigned bits, use `uint8_t` like you did.

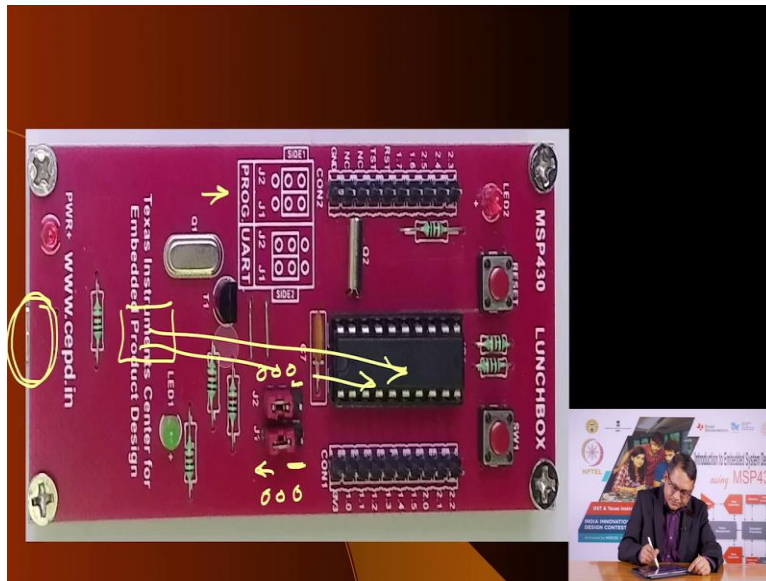
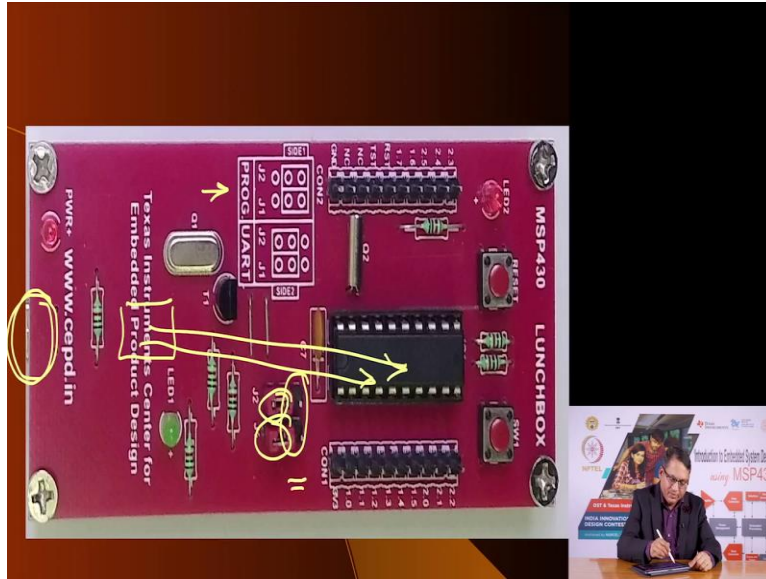
These are cross-platform. You can compile your program on a 32-bit or a 16-bit controller, and you won't have to change the types.



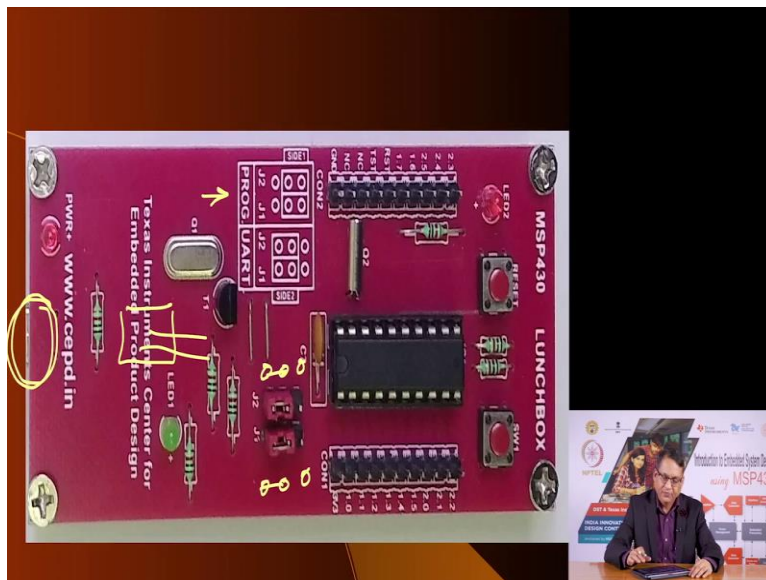
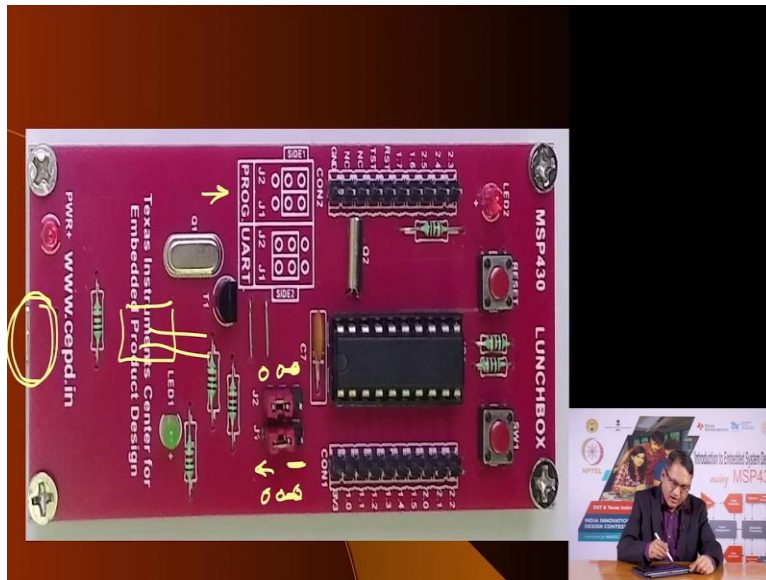
Of course we can, there are many ways as I mentioned, usually it would suffice to have `int` and `char` and maybe a `float` type of variable, but often times you can use `uint` or `uint16` so you are defining that it is an integer but it is of 8 bit size or it is an integer of a 16 bit size. Of course you need to know where these header files which define these kinds of data are available. So, if you

are going to use this, you have to include int types to attach into the C program that you wrote, write.

(Refer Slide Time: 48:17)







Well, now coming back to the whole point of using a serial debugger or serial print, here is the micro controller lunchbox that you would have received. Now, if you see here, there are, we are going to use we are underneath here. Here is the u, USB port. And the USB port there is a chip underneath here which is a USB to urt convertor, which connects to appropriate uart pins on the micro controller.

Now, it turns out that there are two uart codes so to say, one is a conventional uart code, that is a serial communication code. And the other is the code to be used when you want to use the USB to download the program through the boot strap loader from the PC into the memory of the, this micro controller.

And so you have to select some jumpers and that is being shown here. When you want to program, when you want to download the program, the jumper should be on this side. And when you want to use this micro controller and you want to use the USB to transfer information from the micro controller into back to your PC, then you must change the jumpers to these locations. So, that is what is the meaning as they are set here.

So right now, if you try to connect this lunchbox to your PC, and compile the code and download it, it will not download. Why? Because it has not made the connection to the appropriate pins. So, be very careful and sure that when you are downloading code, the jumper should be placed here. Let me show it again here.

When you are downloading code, the jumper should be placed here. And when you, in case you want to use the serial port to the micro controller needs to use the serial port to communicate to the PC while it is running, while your program is running, then you need to pull the jumpers from here and engage in the these two holes.

So, there are three holes here, 1, 2, 3 and 3 holes here, 1, 2, 3. 1, 2, 3. So, you would short these and this when you want to do programming. And if you are going to be using the serial port, then you are going to, you have three pins here again, three here. You are going to short this and this and this and this. And then it would connect the uart pins to the USB, to uart convertor and it will allow the PC to receive values from the micro controller.

Now, once you have the code in the C compiler, you have to go and build the project. And you say build all, and you do that rebuild, then it will not only compile, it will also download. Why? Because you have already instructed that once the code is compiled what is to be done with that code. That it has to be downloaded through the USB using the boot strap loader facility into the micro controller that is connected.

(Refer Slide Time: 51:15)

## Structure of Embedded C code

```
1 #include <Library/LunchboxCommon.h>
2 #include <msp430.h>
3 #include <stdio.h>
4 #include <string.h>
5 #include <inttypes.h>
6 /*
7  * An example of multi-line comments
8  * @brief entry point for the code
9  */
10 int main(void)
11 {
12     //! Stop Watchdog (Not recommended for code in production
13     //and devices working in field)
14     WDTCTL = WDTPW | WDTHOLD;
15     initialise_SerialPrint_on_lunchbox(); // a function
16     int x = 0;
17     while (1)
18     {
19         x++;
20         printf("Hello world %d \r\n", x);
21     }
22     int i;
23     for (i = 0; i < 20000; i++);
24 }
25 }
```

And so, when you do that, if you were to compile this program that we were discussing earlier, here we have compiled this and downloaded it. After you have downloaded it, you change the jumper setting to connect to the uart. This is, you would start seeing the values on your PC. But I missed a point, that for the PC to show the values that are coming from the micro controller, you need to have a programming running.

(Refer Slide Time: 51:46)

## Download and Install PuTTY

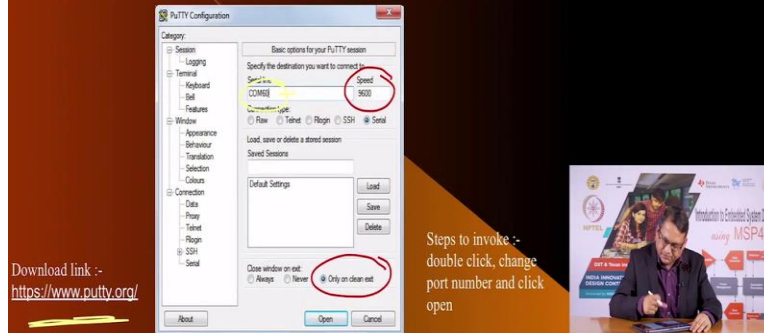
PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. PuTTY can be used as a serial terminal for this Code Example.

Download link :- <https://www.putty.org/>

Steps to invoke :- double click, change port number and click open

# Download and Install PuTTY

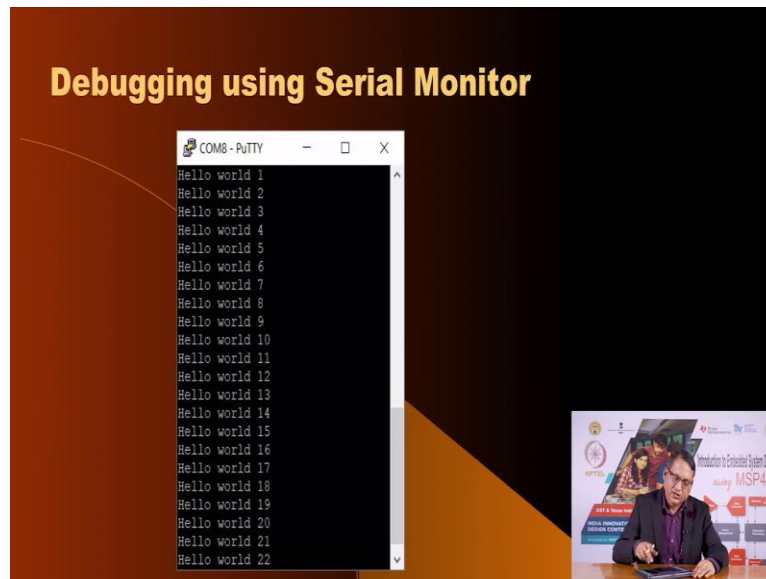
PuTTY is a free and open-source terminal emulator, serial console and network file transfer application.  
PuTTY can be used as a serial terminal for this Code Example.



And that program is called PuTTY. It is a serial terminal emulator. You can download it from this link. Now in this also, because there are many USB ports, you could be communicating through any USB port. You need to tell the PuTTY configuration screen which code you are going to use.

So, here in this case for example it says, COM60, of course not. You find out from the device manager which comport your lunchbox is connected to and you write that number. And then the serial communication speed has to be set to 9600, because our micro controller is going to be sending values at this rate. What this rate is and what does it mean, we are going to cover when we do serial communication in one of those lectures. And then you do only on clean exit and then you open this, alright?

(Refer Slide Time: 52:41)



And then, when you do it, and your program is running. You have already downloaded the program and when it is running on your micro controller, this PuTTY window will open and here is what you see. Hello world 1, 2, 3 and you will see every second or so, this value is being updated. So, this is the way serial debugging could be implemented with the micro controller. Now, once you are happy with the way it is working, you can use this feature later on and maybe in one of those lectures we will, one of the subsequent lectures, we will show, how to use this serial monitor facility to print intermediate information on your PC for debugging purposes or for any data, you know, monitoring purposes.

Right now, let us come back to using the C programming language for issues related to Embedded C programming. And one of the things that it needs to do is to manipulate bits because it is going to get information from the PC, from the various ports or value stored in the memory and then it needs to decide whether the number is 0 or whether a particular bit in this number is 1 or shift that bit certain bits to the left or to right, so you need to know instructions which allow you to manipulate these bits.

(Refer Slide Time: 54:08)

## Bit level access tricks on registers using C


```
• Set Bit: ✓
PIDIR = 8; //0000 1000
PIDIR |= (1 << 2); //0000 0100
PIDIR |= (BIT7 + BIT6) // 1100 0000
→ Result 1100 1100

• Clear Bit:
PIDIR = 204; //1100 1100
PIDIR &= ~(1 << 3);
→ Result 1100 0100

• Flip Bit:
PIDIR = 8; //0000 1000
PIDIR ^= (1 << 5);
→ Result 0010 1000
```

```
• Get Bit: ✓
PIDIR = 8; //0000 1000
if ((PIDIR & (1 << 3)) > 0)
{
    // do this if the 3rd bit is set(1)
}
else
{
    // do this if the 3rd bit is not set(0)
}
```

$$\begin{array}{r} 00001000 \\ 00000100 \\ \hline 00001100 \end{array}$$



And so instructions which allow you to play with these bits is, are called bit manipulation instructions. And so we say, for example, you want to set a particular bit in a variable, so let us say the variable, although this is not really a variable, this is a port that we are going to talk about once we talk about the digital input and output ports of micro controller. But imagine that this port, this could be as well this p wonder could as well be a another variable. And if that is initialized with the value 8, now 8 in decimal is also 00001000.

And so, we want to set certain bit, what do you want to do? We want to set the, so this is 0, 1 and 2. We want to set the second bit. So, the way to do that is we want to set the second bit without affecting the other bits. And so what we do is, we, we did that variable or with the variable which is shifted, in which 1 is shifted two places to the left. And so once you or it, this 8 is going to be or with the 0000100. And so the result will be 00001100. This would be, this you would get at this point.

Now, you want to take the bit 7 plus bit 6 that means you want to set this and this bit and you want to or it with the original value. And so, when you execute all these three instructions, initial value loaded into that variable was 8. But after these two additional instructions that is this and this, you would end up with this result. So, please go through this code and satisfy yourself that that is what it is going to happen. So, this is a mechanism to set arbitrary bits in a, in a variable.

In this case, this variable happens to be a port address or a port location. Then you can clear certain bits, that is you can turn certain bits off, and for that you use And function and you And it with the invert of that, so that particular bit becomes 0. So, you have taken 1 and you have shifted it three digits to the left. And then you invert that so that particular bit becomes 0, rest all become 1. And so when you and it, the other bits remain unaffected. The bit 3, that is bit 0, 1, 2, 3, bit 3 is made 0. It was 1 it will become 0. It was 0 it will remain 0.

Then you can flip bits using this these set of instructions. A particular bit that is bit number 5 and when I say 5 it means 0, 1, 2, 3, 4, 5, that bit needs to be flipped. That is if it was 1, it becomes 0. If it is 0, it becomes 1. And so, as you see, the number is 8. In this, this bit is 0, 1, 2, 3, 4, 5. This bit is 0. When you execute it, this bit becomes 1. This is the operation to flip bits. And here you can get a bit, that is you want to know a particular bit is the value 1 or 0, if it was 1, you would do something, if it was 0, you do something else.

This set of instructions will allow you to do it. How? You have initialized it with some value, now you want to know if that a particular bit is 1, it will do something, if it is 0, it will do something else. You And the original value with 1 shifted 3 bits to the left. And if the ending operation is 0, you do something. If it is non zero, that is it is greater than 0, then you do something else here. And so these are the mechanisms to play with various bits.

(Refer Slide Time: 57:53)

The image is a composite of three parts illustrating bit manipulation in C:

- Slide (Left):** Titled "Example of Embedded C: demo of Bit Manipulation". It shows a terminal window with the following output:
 

```

x value 00000010
y value 00000100
z value 00000100

Bitwise operators in C

value of a & b 00000010
value of a | b 00110010
value of a << 1 01000100
value of a * b 00110000
value of 1 << 3 000001000
value of a | (1 << 4) 001010100
      
```
- Code Editor (Middle):** Shows the source code for the demo:
 

```

1 #include <msp430.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include <inttypes.h>
5 #include <library/LunchboxCommon.h>
6 int main(void)
7 {
8     // Stop watchdog (Not recommended for code in production
9     // and devices working in field)
10    MDCTL = MDTPW | MDHOLD;
11
12    initialise_SerialPrint_on_Lunchbox(); // a function
13    while (1)
14    {
15        uint8_t x = 2;
16        uint8_t y = 4;
17        uint8_t z = 8;
18        printf("x value: %d\n", x);
19        decToBinary(x);
20        printf("y value: %d\n", y);
21        decToBinary(y);
22        printf("z value: %d\n", z);
23        decToBinary(z);
24        printf("Bitwise operators in C\n");
25        uint8_t a = 0b01000100; //58 or 0x5d
26        uint8_t b = 0b00100100; //36 or 0x24
27        printf("value of a & b: %d\n", a & b);
28        decToBinary(a & b);
29        printf("value of a | b: %d\n", a | b);
30        decToBinary(a | b);
31        printf("value of a << 1: %d\n", a << 1);
32        decToBinary(a << 1);
33        printf("value of a * b: %d\n", a * b);
34        decToBinary(a * b);
35        printf("value of 1 << 3: %d\n", 1 << 3);
36        decToBinary(1 << 3);
37        printf("value of a | (1 << 4): %d\n", a | (1 << 4));
38        decToBinary(a | (1 << 4));
39        printf("\n");
40        unsigned long delay;
41        for (delay = 0; delay < 60000; delay++);
42    }
      
```
- Video Frame (Right):** A small video frame showing a person presenting, with a book titled "Introduction to Embedded System Design using MSP430" visible in the background.

You, you could you know, look at this code which is available in the repository. Here it does lot of operations, you could modify them and since we have included the serial print on the lunchbox, you would be able to see the result on the serial monitor and then you can see what it is doing, you could vary numbers here, change numbers here and rerun the program, re compile, rebuild and download and you will see, you can check your understanding of these instructions by predicting what number should come and see whether that number is coming.

(Refer Slide Time: 58:27)

Blink led example using msp430.h

```
1 #include <msp430.h>
2 #define entry_point_for_the_code*/
3 void main(void)
4 {
5     WDTCTL = WDTPW | WDTHOLD;
6     /*(volatile unsigned int *) 0x0120 = 0x5A00 | 0x0000;
7     /* ! Stop watchdog (Not recommended for code in production
8     and devices working in field)
9     */
10
11     P1DIR |= BIT7;
12     /*(volatile unsigned char *) 0x0022 |= 0x00;
13     // P1.7 (Red LED)
14
15     volatile unsigned long i;
16     while (1)
17     {
18
19         //PIOUT |= BIT7;
20         PIOUT |= 0x80; //Red LED -> ON
21         /*(volatile unsigned char *) 0x0021 |= 0x80;
22         for (i = 0; i < 10000; i++)
23             ; //delay
24
25         //PIOUT &= ~BIT7;
26         PIOUT &= ~0x80; //Red LED -> OFF
27         /*(volatile unsigned char *) 0x0021 &= ~0x80;
28         for (i = 0; i < 10000; i++)
29             ; //delay
30     }
31 }
```

Now, a lot of things that you would do on the micro controller would deal with ports, read the ports, manipulate the value that you get from the ports and output into some other ports, alright. And so here is an example, which if you would run, right now, you would see a bit connected on the LED connected on the lunchbox would blink at certain rate. For that, we have to tell the micro controller that such a bit, such a LED is connected to such a port which has a certain name.

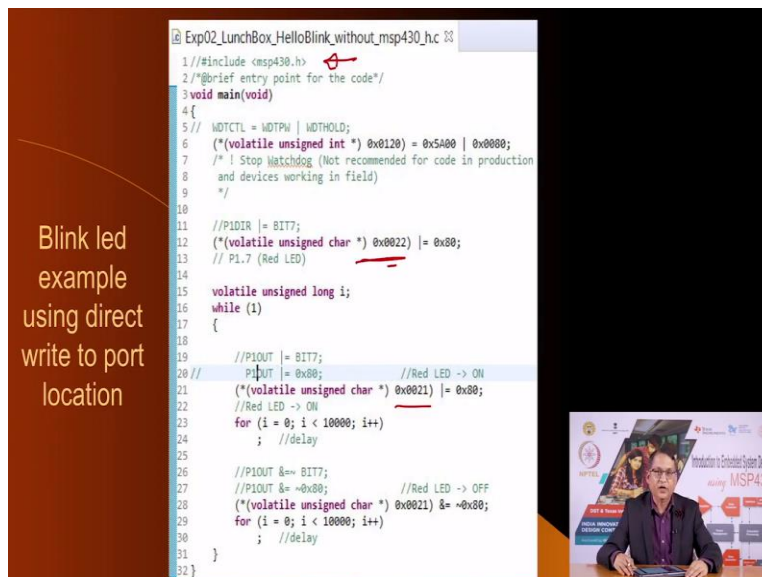
And as we have, as we will see when we deal with the MSP430 architecture that these ports on MSP430G2553 micro controllers, there are two ports, one is called p1, the other is called p2. And in our case, we have bit 7 of port 1, we have connected a LED. And so, we are dealing with those ports using a name, these names are defined in this header file. And so, we do not have to remember the addresses of those ports, we simply refer to them with these names.



So, it is like somebody has already declared a variable whose name is this and you can deal with these names, manipulate information in these locations. But it is possible to access those locations as memory addresses also, because you see, MSP430 is a von human architecture, which means the micro controller or the processor inside has a unified memory map, and in the same memory map, you are going to store the program, some other parts of that same memory map are going to be used to store variables, because you have RAM there. And some other locations are port addresses.

So, you could look at a port 1 as a name available through this header file, or you can find out, what is the address of port 1 and directly access that memory location. So, the same program I would show that you can get rid of this, you can remove this hash include, but then you would not, the C compiler would not recognize this P1 dir if you used it, or p1 out if you used it, it would not know. So, you need to replace it with addresses of these two ports.

(Refer Slide Time: 60:38)



```
Exp02_LunchBox_HelloBlink_without_msp430_h.c
1 // #include <msp430.h>
2 /*@brief entry point for the code*/
3 void main(void)
4 {
5 // WDTCTL = WDTPW | WDTHOLD;
6 (*(volatile unsigned int *) 0x0120) = 0x5A00 | 0x0000;
7 /* ! Stop Watchdog (Not recommended for code in production
8 and devices working in field)
9 */
10
11 //P1DIR |= BIT7;
12 (*(volatile unsigned char *) 0x0022) |= 0x80;
13 // P1.7 (Red LED)
14
15 volatile unsigned long i;
16 while (1)
17 {
18
19 //P1OUT |= BIT7;
20 // P1OUT |= 0x80; //Red LED -> ON
21 (*(volatile unsigned char *) 0x0021) |= 0x80;
22 //Red LED -> ON
23 for (i = 0; i < 10000; i++)
24 ; //delay
25
26 //P1OUT &= ~BIT7;
27 //P1OUT &= ~0x80; //Red LED -> OFF
28 (*(volatile unsigned char *) 0x0021) &= ~0x80;
29 for (i = 0; i < 10000; i++)
30 ; //delay
31 }
32 }
```

And the second example here identical it just replaces with the memory address. In this case it is 22. And in this particular case it is 21. And the same program, we are just, you know commented out the names of those ports and replace them with the addresses. And, and as you see here, we have also commented out the header file. So, we are not using the information in the header file, we are directly writing into those memory locations. Even then the same program works. So, the whole point of this exercise was to illustrate that when you are compiling code, essentially all

these instructions, at the end of the day for a von human architecture, modify certain memory locations.

Whether they are registers or whether they are variables, or they are constants, they are all in the same memory map. If it was Harvard architecture, then of course there will be data memory for which you would need to deal with those data memory in a different way and you would have programmed memory and you would need to deal with that in a different way. In fact here, MSP430 not only it is von human architecture, but also that the ports are, you can see ports in two ways, one is called actual ports and the other is called memory map ports.

In MSP430, these ports are mapped into the memory map, that is the port addresses are like memory locations, which means any port can be seen as a memory location, any memory location can be seen as a memory location. And in fact this the point of this exercise was to show that a port is also a memory location and we can manipulate the information in the case of MSP430. Why? Because it has a concept of memory mapped ports.

Ports which are seen as memory locations. And if they are seen as memory locations, their values can be manipulated by reading and writing to those memory locations. All you need to know is the address of that memory. And this second exercise is actually to illustrate that. Once you know specific and exact address locations, you do not need any names to access those ports or those memory locations. You can directly read and write. And when you read and write in this case, you are actually changing the way the LED toggles.

And so, with this, we cover three important elements of our journey, which is to understand what is version control system, version control mechanism software for doing that. And in this case, we have used GIT. With that execution of that line you have pulled the repository from the GIT hub website and you have created a local repository on your computer and then we install the code composer studio. Then you saw how the code composer studio could be made to point at those folders which contain these code examples.

You could import them and you could build them, or you could modify them. And once you rebuild them, you would be able to download the code into the memory of the micro controller. And then a little bit about elements of Embedded C programming. With this we finish this topic

and I will come back with a new lecture and deal with various aspects of MSP430, architecture and programming, and I will see you very soon. Thank you.