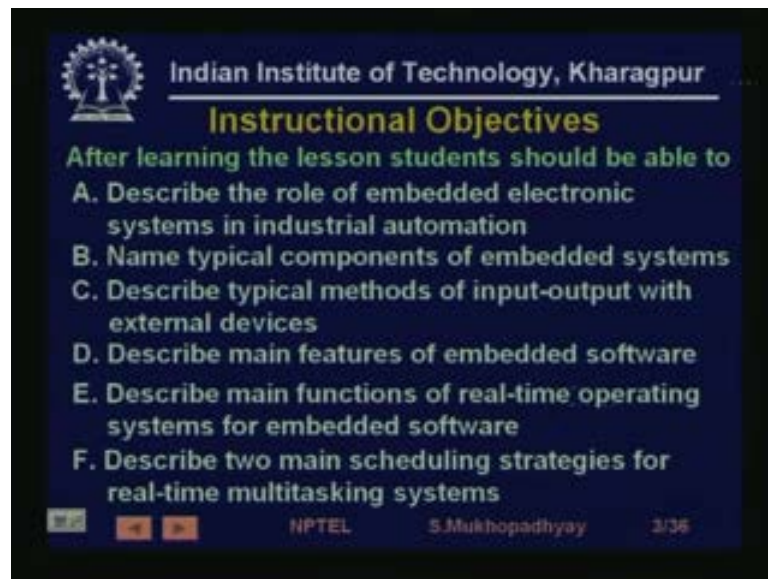


Industrial Automation & Control
Prof. S. Mukhopadhyay
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
Embedded Systems

Welcome to lesson 36 of the course on Industrial Automation and Control. In this lesson, we are going to look at Embedded Systems, which are basically programmable electronic systems, had wide applications in various kinds of technologies, including industrial automation. So, lot of industrial automation products and systems are finding embedded systems in there them. Therefore, this lesson is meant to give a basic introduction to embedded systems, for industrial automation, how they work, what they contain, what is the basic principles, etcetera.

(Refer Slide Time: 01:44)



The slide is a dark blue rectangle with white and yellow text. At the top left is the IIT Kharagpur logo. To its right, the text reads 'Indian Institute of Technology, Kharagpur' in white, followed by 'Instructional Objectives' in yellow. Below this, a green line of text states: 'After learning the lesson students should be able to'. This is followed by a list of six objectives (A-F) in white text. At the bottom, there are small icons for navigation and a footer containing 'NPTEL', 'S.Mukhopadhyay', and '3/36'.

Indian Institute of Technology, Kharagpur

Instructional Objectives

After learning the lesson students should be able to

- A. Describe the role of embedded electronic systems in industrial automation
- B. Name typical components of embedded systems
- C. Describe typical methods of input-output with external devices
- D. Describe main features of embedded software
- E. Describe main functions of real-time operating systems for embedded software
- F. Describe two main scheduling strategies for real-time multitasking systems

NPTEL S.Mukhopadhyay 3/36

So, the instructional objectives are the following, first the role of embedded electronic systems in industrial automation, how various products and systems and equipment and machines are using them. Then, delving into the nature of the technology, what are the typical components in it, what are that typical methods in which these embedded systems are for industrial automation, they are basically computing systems which interact with the outside world, so high.

What are the basic methods of communicating with external devices; they are programmable processor base system, so they contain software. So, some of the basic main features of embedded software and this software's, after a certain complexity, you have the software's run under the umbrella of another software, call the operating system.

See in this case, you use a real time operating system, so give your introduction to that, and finally, there are many things, that a device needs to do everything in proper time. So, scheduling that is, when which computation will go on in the system, these determining that is actually very important. So, this called scheduling and we will look at to typical scheduling strategies, used in such systems. So, first take a look at how an embedded system is becoming very important for industrial automation.

(Refer Slide Time: 03:28)



A typically production technology has involves numerically controlled machines, they are computer numerically controlled; sometimes they are directly numerically controlled from a computer, many machines using some communication mechanisms. Then, there is a process control, if you go to the process control, big process control at process automation company sites, you will find that they advertise.

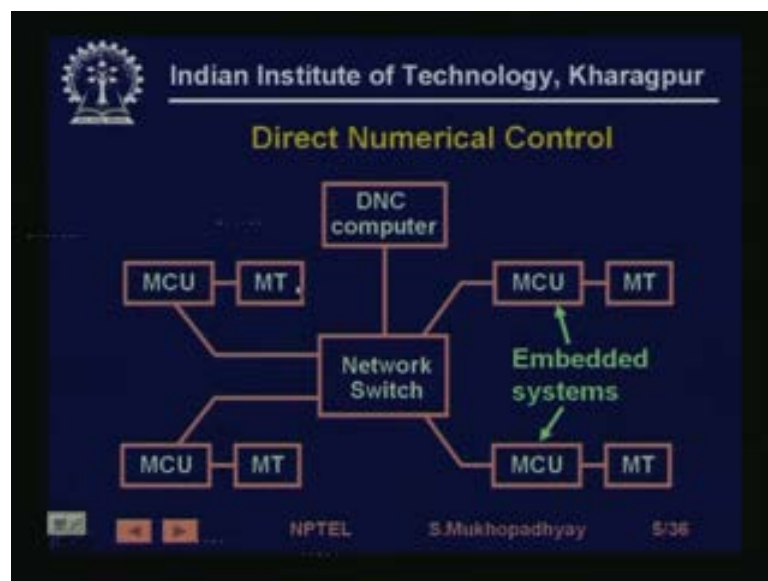
Apart from the conventional things, they advertise a lot of embedded system components, like smart actuators, smart sensors and intelligent distributed process controlled systems or all these use embedded system. Then, they in manufacture in

people use vision systems, camera based systems for inspection, for safety operations, they use robots.

Automated material handling systems, like automated guided vehicles, automation of heavy movements, like cranes some things like that. So, each one of these involves contain embedded systems in them. Finally, today we talk about flexible manufacturing system which is nothing but a system, which causes coordination between such machines, which are coat and un coat intelligent.

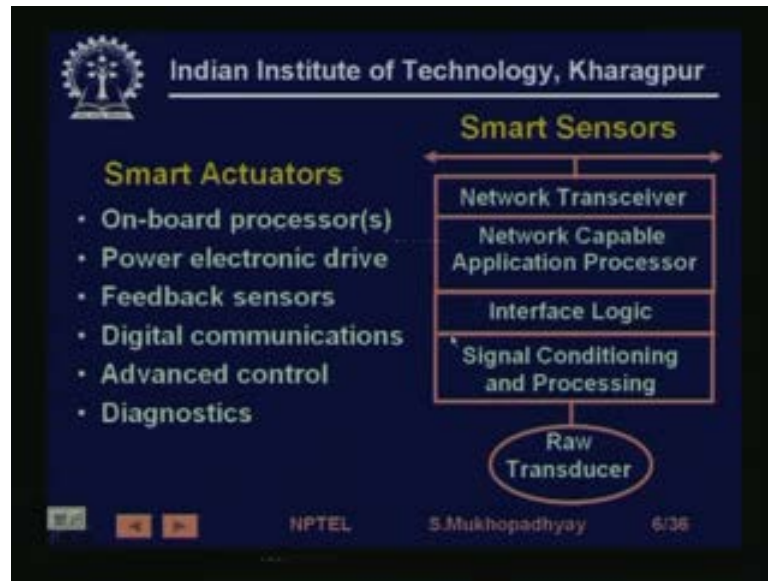
Means that they can they can examine signals, they can look at there in one and they have some amount of computational intelligence and so coordination is possible. And therefore, through coordination, better productivity, better efficiency is possible in such systems.

(Refer Slide Time: 05:16)



So, enabling, if you take a look at that, say DNC something which is , so here you have a DNC computer and these are the individual machines, so MT means a machine tool and MCU means the machine controlled unit. So, the machine control itself is an embedded system, which will typically today contain more than one microprocessor based electronics. And in a direct numerically controlled system, the several machines are controlled from one computer through a network, so these MCU unit is are typically embedded systems.

(Refer Slide Time: 06:03)



Similarly, if you look at smart actuators and sensor, they are all for example, today you have normal process control valves and you have smart valves, which can do first of all which contains loops, which have I mean digital controllers for positioning valves, drives, power electronics, not only that it has software, which can do online calibration and it can do diagnostics, etcetera, it can signal failures.

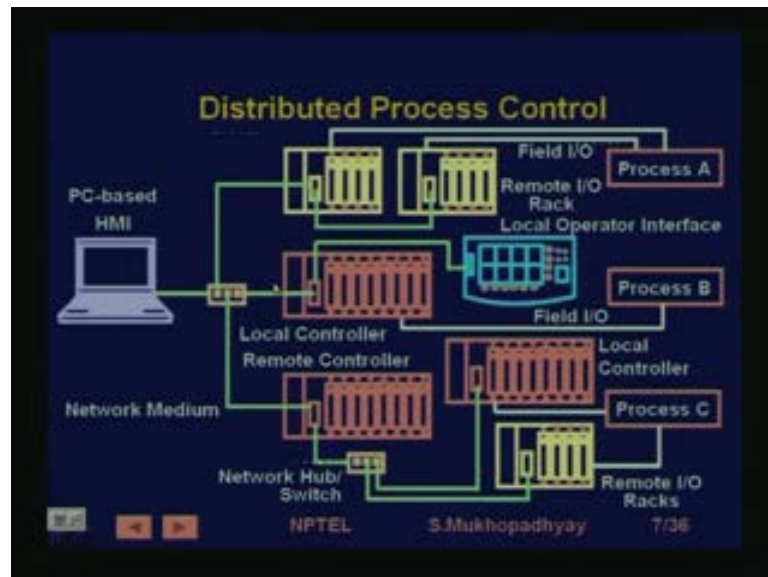
So, all these are possible, because of the presence of embedded electronic systems with the valve, which are field mounted and now in directly present on the valve itself. Similarly, you can have smart sensors as we will see that, we are going to have sensors, which can directly connect to digital communication networks and can transmit there values to a to a remote operator interface or to a remote controller.

So, such smart sensors the typical architecture is shown, so you had you have the real transducer may be let say, it may be an RTD and there this may be the signal conditioning and processing circuit for RTD, which may be a bridge. And then this analog signal out of the bridge is going to be interfaced to some application processor, which will do two things.

Firstly, it will take this signal and then may be do some signal processing and may be put it into may be to some linearization and other things digitally for the signal processing. And then finally, that is one side of it, the other side of it is there, it will be able to transmit, these values on the network. So, it will make this net, this whole sensor or the

RTD will become network compatible, in the sense it can directly communicate, it is readings into the network. So, such a thing is called as smart sensor and is basically possible, basically this part of it is the conventional sensor, after that you have this embedded system, which actually makes it a smart.

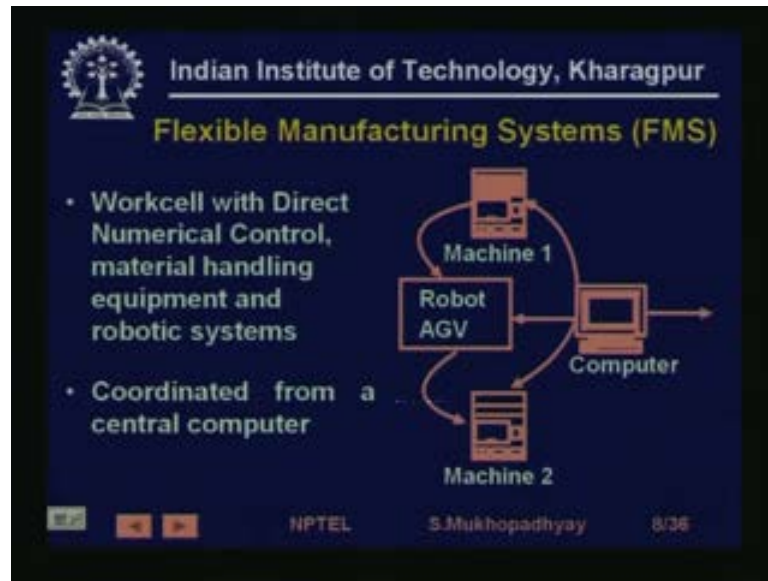
(Refer Slide Time: 08:16)



Similarly, if we you have distributed process control system solutions, so there you have, you can have a various kinds of architectures. Here is an architecture, where you have, field IO connected to remote IO racks and this remote IO racks will take a number field IO channels and then we will transmit then over network. So, all the green lines are actually computer communication networks.

And similarly, they may communicate to some local controller or sometimes, the local controllers make an in turn communicate to a remote controller. So, each one of these things actually contain a various kinds of cards, which are actually microprocessor based cards and which are embedded which have some embedded computational electronics in them. So, they make possible the coordination and transmission of variables, I mean the large number of amount of data can be made available to all over the place in the factory for possible use in for improved coordination and efficiency, etcetera.

(Refer Slide Time: 09:41)



So, similarly when you are flexible manufacturing systems, flexible manufacturing system essentially it is coordination, so you have all these embedded systems. So, basically, you see that, once you have local systems, which have computational probability in them, because of the embedded electronics. Then one can easily think of the next, would be think of communication of digital electronic communication between these and so called smart devices.

So, that they can coordinate amongst each other, for improved performance, so flexible manufacturing system is exactly that, so you have a concept of work cell, where you have this and this. Computer control machines, you have these, I mean smart automated material handling systems, you have robots and depending on the batch, that you had processing, they can be always reprogrammed and they can do a various kinds of function in very efficient manner, under the coordination of a computer, so that is called flexible manufacturing.

(Refer Slide Time: 10:52)



Indian Institute of Technology, Kharagpur

FMS Components

- Vision Systems**
 - Real-time video image processing and decision technology. Used for inspection
- Automated Material Handling**
 - Coordinated actuation system to move parts and equipment in manufacturing.
- Robots**
 - Coordinated actuation system to achieve various manufacturing tasks such as welding, assembly, paint etc.

NPTEL S.Mukhopadhyay 9/36

So, what we see is essentially that, similarly these manufacturing system components used vision systems, automated material, handling robots, some things like that.

(Refer Slide Time: 11:00)



Indian Institute of Technology, Kharagpur

Trends for Industrial Automation

- Intelligent Sensing, Control and Actuation
- Communication
- Real-time Decision
 - Monitoring, Supervision, Coordination, Diagnostics, Optimisation

Enabling Technology : Embedded Computing Systems

- Integration with Planning and Management
 - CAD, Process Planning, Scheduling, Inventory Management

NPTEL S.Mukhopadhyay 10/36

So, basically if you see, then strong trends in industrial automation is, firstly to have more devices, which can perform devices for sensing, actuation as well as control, which can do, which are more sophisticated, which use more complex control algorithms, which can do, which can monitored themselves. Calibrate themselves, diagnose their

own health, etcetera, which are all possible basically by in situ computation, using on device electronics, computing electronics processors.

So, which are basically called embedded systems and then so these enable you to take real time decision and these devices are also communication enabled. So, once you have these, then locally you have a lot of intelligence and lot of computational task can be performed locally. And then finally, once you have these smart systems placed or all over the factory, then it becomes very simple to integrate them.

Both in the horizontal level across the factory as well as vertically from between planning and design to manufacturing execution systems down to the automatic control levels. So, you have a vertical integration, which we have seen in the automation pyramid as well as you can have very great horizontal integration, using network from that is all over the factory, at all the levels can be integrated, which is possible through networking.

And in the next two lessons of this course, we are going to talk about, such an industrial communication network called field bus. So, as we see that basically in these lessons we will be talking about embedded systems, we which enable lot of real time decision making and intelligence sensing actuation control. And these in turn act as enablers for integration using communication technology.

So, in the next three lectures, in this lessons we are going to learn about embedded systems, in the next two lessons, we learn about one industrial communication system called field bus.

(Refer Slide Time: 13:30)

Indian Institute of Technology, Kharagpur

Brief Introduction

- **Embedded systems**
 - 'Special purpose computers built into a larger device'
- **'Special-purpose'**
 - Application-specific
 - Both hardware and software tailored to application
- **'Built into a larger device'**
 - An ES is a part of a larger device, interacting with it to enhance its capabilities

NPTEL S.Mukhopadhyay 11/36

So, what are embedded systems definition wise, there are a variety of definitions, but very simply stated, an embedded system is a special purpose computer built into a larger device, it is called embedded. Because, for example, if you look at this, you are not probably able to see, but let us say, this computer screen and I have a PC. Just below me, whom you cannot see, so this PC is not an embedded system, because it is a stand along computing system.

It is a system only for computation, but for an embedded, but imagine a processor, which controls let us say to give you a very simple example a washing machine. So, that is also a computer, there is also a processor, there is memory, s in principle that processor and this processor are quite similar. But, that is called an embedded system, because first of all, that is a very special purpose computer, that is built just for controlling the functions of the washing machine and it is built into the washing machine.

So, it is embedded inside it, so you cannot as such see it, unless you really open it out and it works from within a larger device. So, we call it special purpose, because it is built solely for the further purpose. While, this PC is actually a general purpose computing machine and could be used for a variety of functions, including control, various kinds of displays, storage, general computation, communication.

So, PC's have been used for all kinds of functions, while these are very specific application, specific devices and it is a part of larger device. So, that is a very general and simple definition of an embedded system.

(Refer Slide Time: 15:18)



What do they contain; they are actually processor base systems. So, they generally contain a number of processors is there, they can be various kinds of, they can be micro controllers, they can be micro processors, they can be special purpose processors like digital signal processors, etcetera. Obviously, contain program and data memory all microprocessor base systems contain that and they contain special circuitry.

Because, these are systems, which are typically built for interacting or responding to the environment, so they get lot of signals from the outside world and they also give out lot of signals, so the outside world. So, input output is very important for them and input there are these outside world signals when they will come and they are that is not known. So, therefore they have different mechanisms of communicating interrupt is one of the mechanism between C.

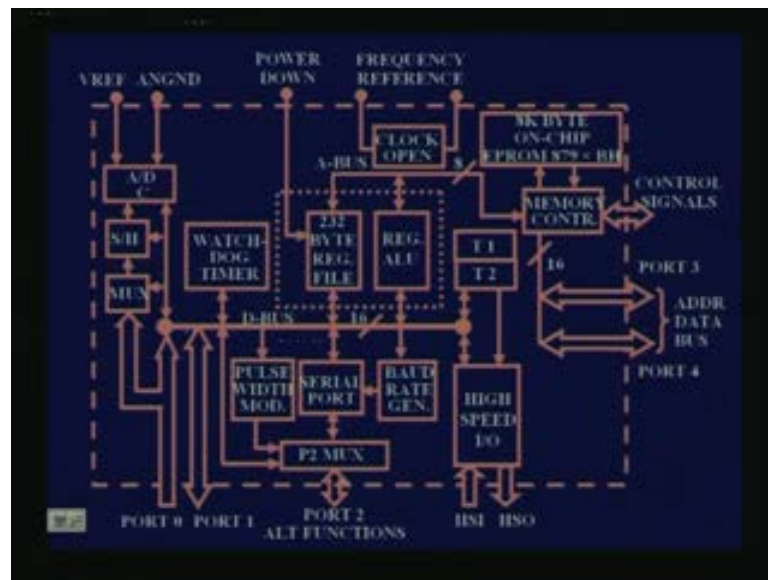
Time is very important in most of the systems, because if many of these control tasks are actually time oriented, once they start they have to be finished within this time or once somebody demands, some kind of value with just to be given within sometime. So, there all some timer circuitry required, similarly since they have lots of communication

requirements. So, they are generally vary, support some communication interfaces, where sometimes very critical components in systems.

So, they have to be tested and they provide testing the phases, sometimes they require human operator interfaces like various kinds of displays. So, they want they need human interface, device interfaces, these are the kinds of interfaces that they have, they have certain some applications specific ICs, which is called a 6.

Sometimes they called contain reprogram able ICs like SPGS, CPLDs and things like that, things were the IC can be programmed by software, to behave like a particular applications specific hardware. They; obviously, contain buses, some of the buses are special purpose buses, particularly designed for specific embedded applications and in terms of software, the contain a real time operating system, running in them and finally, they contain the application program, so on the tasks. So, if you take a typical embedded system, you will find many of these, you many not find all, but you will find many of these in an embedded system.

(Refer Slide Time: 17:56)



For example, let us say this a particular 8096, this a slightly old processor, which was designed for embedded system applications and if you see it, you will see that, it contains many things which are required for embedded application. Like for example, it contains this is the processor, this dotted part is actually a processor, it contains this basic processing as well as it contains a resistor file.

So, in the see it contains analog to digital converters and analog multiplexers, this chip can interface with analog signals, it contains lot of ports, it contains a serial port, it contains parallel ports, it contains timers, see timer 1, timer 2 and watch dog timer. It contains on chip memory that is at least for building, small scale applications, you do not have to interface, this with external memory.

But, you have also been provided, facility to interface with external memory, if you need to through these ports. In fact this system is somewhat configurable in the sense that, if you need memory, this can work as memory interfaces, address and data buses, if you do not need them, then they can just work as ports. Similarly, if you want this chip to drive, some power electronic actuator, then you often need a very clean pulse with modulation signals.

That, we have seen in the as a power electronics drives that many of the motor drives actually require pulse with modulation signals. So, this has a pulse with modulation circuitry on chip, which can actually generate the signal through this port. So, you see that, this a typical process and nowadays you have higher, these processors can be of various complexities in the embedded system market, you have processes from 8 bits, 16 bits, 32 bits. So, you have lots of various kinds of processors, so this is a typical an embedded system process.

(Refer Slide Time: 20:32)

Indian Institute of Technology, Kharagpur

Constraints and Performance goals

- **Constraints**
 - CPU, Memory
 - Power consumption
 - Limited peripherals and slower buses
 - Size, weight, environmental reliability
- **Performance goals**
 - Latency Bounds
 - High Reliability
 - Interoperability

NPTEL S. Mukhopadhyay 14/36

Now, if you embedded systems typically for various reasons, I mean actually embedded system for industrial automation is somewhat well behaved. In the sense, that embedded systems are required to be sometimes working under there are various other kinds of embedded systems, which other than embedded system for industrial automation. Like, let us say, if you have a digital camera, that also contains a processor, that has an embedded system.

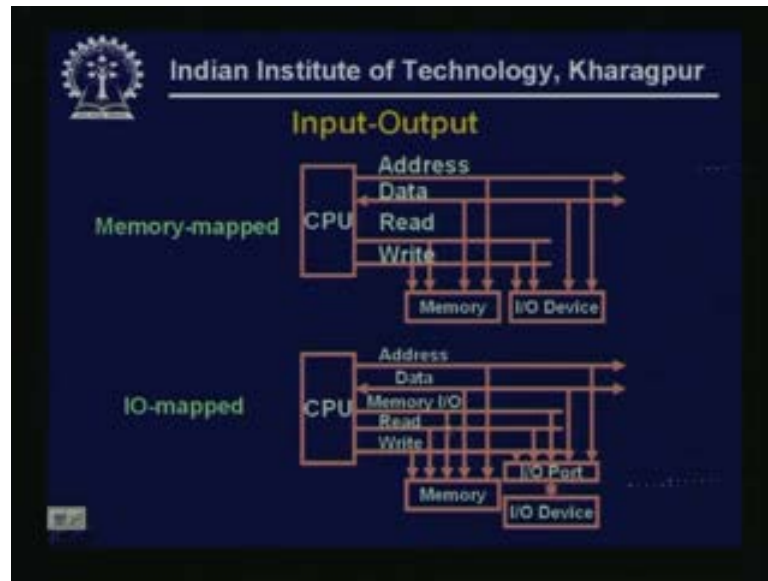
If you have a watch, it can have an embedded system, so many times, these systems actually work under constraints of CPU power, in the context of industrial automation people generally use speed may or may not be such a constrained. So, people use, often use chips, which do not have very high clock speed, because in the name of reliability. There may be memory, may be scarce, power consumption is often very important consideration.

Because, if you have to continuously replace many of the systems could be are battery operated. And therefore, to in to increase battery like power consumption is very important. Sometimes they do not have limit, so many peripherals and they have slow buses, then since some of the systems actually must be very small. Since, they actually have to embed themselves in small, small devices, so they work with size, weight and some time environment reliability constraints.

On the other hand, they have very big performance goals, like first of all they have latency bounds, since they controlled events. So, it is generally preferred that there response times to the signals that they receive from environmental actually is very, very important. And it is often desired that, these response times are bounded, they need to work with very high reliability, because they often control machines, where if the malfunction, they can create sometimes havoc.

And they must be interring operable, in the sense that, they have to a work with variety of various kinds of devices manufactured by various types of companies. So, they have to must work with certain, must follow some common standards of operation. These are typical requirements, which are embedded which embedded systems have.

(Refer Slide Time: 23:08)

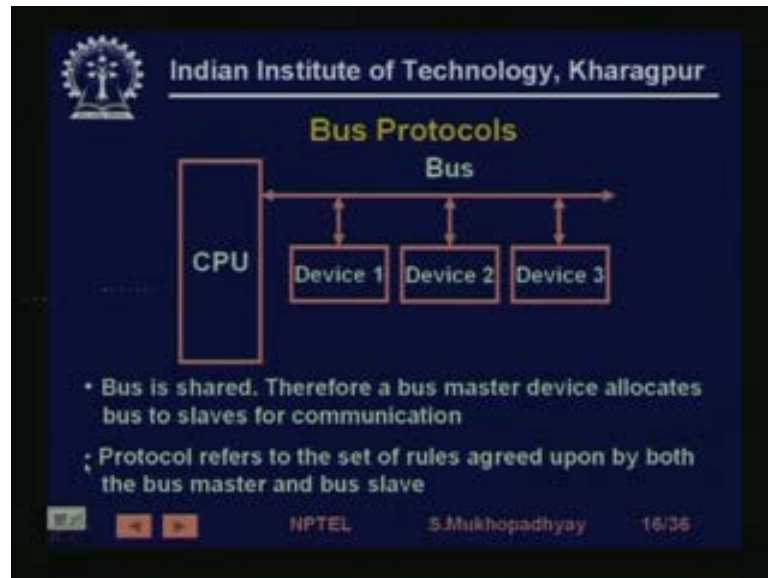


Now, I thought that, we should take a look at the basic principles of input, output, which the embedded step systems support, because input output is a very important functions of embedded systems. And as we know, those who have studied micro process will know, that input output devices, isolated devices can be interface to the CPU in generally two ways, either it looks to the computer, like a memory device.

So, it simply just like, it reads a data from memory, it can read a data from an IO device resistor simply like a memory read or a or a memory write. Or sometimes, they are distinguished, that is memory device an IO device that distinguishes and their address bases are also different. So, whether it is a memory read, so memory read writes and IO read writes are sort of separated out and they are distinguish by this M memory IO line.

So, may be if it is high, then people think it is memory, if it is low, if people think in is it is an IO port. So, in both cases, this kind of input output requires that the CPU itself has to read write data. So, they are generally used when small volumes are data to be transfer from devices infrequently, like AD converter reading, so at a time you get only one reading, so an AD converter generally interfaced in this manner.

(Refer Slide Time: 25:03)



On the other hand, there may be other situations, where devices need to transfer large amounts of data, so or they may be external to the main processor system, in which case people use bus based input outputs, so have a bus to which you connect all the devices. So, now this bus which is nothing but a set of conductors, electrical conductors, basically some lines, all the devices are connected to the same buses. So, therefore obviously, a bus is a shared resource.

So, reading writing means what, reading writing means applying voltages to these lines on the bus, if everybody simultaneously starts applying voltages on these lines, then all these voltages will interact and all the data will get destroyed. So, therefore they must be somebody, who has to decide that, when who will talk, who in use the bus at what time. So, therefore, there is there is usually bus master, who will allocate the bus to various others, who are called just to distinguish them, they area called salves for communication.

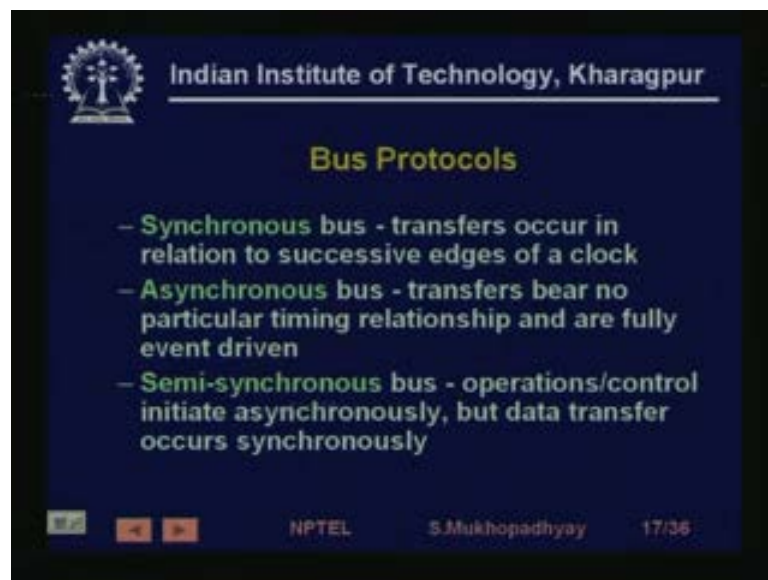
And similarly, for once, since they have to communicate and then everybody is connected, so therefore, they are has to be ways of knowing, ways of telling, that what this is communicating for who is it for. So, there are elaborate rules by following which this communication takes places between the transmitter and receiver, so they are called protocols.

So, in many cases, while some of the devices the CPU directly interacts with them, as if it is a component of its own systems. Sometimes, it will for other devices, one need to set up an external bus, according to some standard protocols, like say Emba is a bus, which is used for embedded systems. So, some protocols and then they connect people make devices according to those protocols.

So, people can put together embedded systems, where the CPU and the devices follow such protocols and communicate. If you extend this concept, you can also have networks, where the buses are actually geographically separated and here, still this bus will possibly be housed within some mechanical cabinet or something or may be some slots.

But, if you extend the same concept, then this bus can become a network bus and then it will be traveling for kilometers and in the same way devices will you connected to them and they will also communicate according to some protocols. So, basically bus base communication is actually very common.

(Refer Slide Time: 28:00)



Indian Institute of Technology, Kharagpur

Bus Protocols

- **Synchronous bus** - transfers occur in relation to successive edges of a clock
- **Asynchronous bus** - transfers bear no particular timing relationship and are fully event driven
- **Semi-synchronous bus** - operations/control initiate asynchronously, but data transfer occurs synchronously

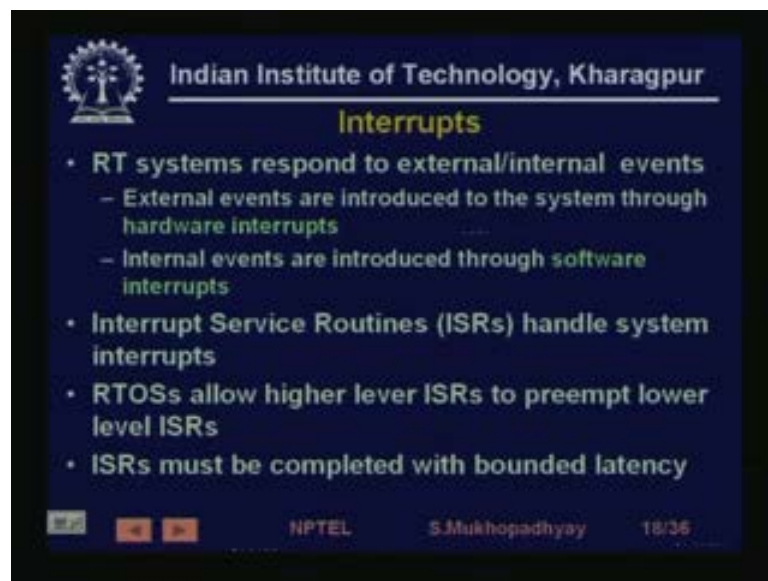
NPTEL S.Mukhopadhyay 17/36

Now, bus protocols typically communicating are typically or either synchronous or asynchronous, in the sense that sometimes, they exchange of data occurs in distinct steps of times, according to some clock. That is useful, when in that way, when a large volume of a regular transfer of data takes place and both the sender and the receiver can actually work at the same rate, then in such cases, this synchronous bus transfers are useful.

On the other hand, in some cases, transfers be a no timing relationship, when they are only ventilated So, after somebody raises another one line, voltage on the places, the voltage on another line, then another device reacts and he will read something and then we will place another, may be some data acknowledged signal will pull low. So, all the actions take place with respect to events, which are occurring on the bus and not with respective some well known I mean some well specified clock.

So, such bus transfers are called a synchronous or sometimes they may be semi synchronous, in the sense that some of the control activities may be asynchronous. But, within the control activities, when large amount of data transfer takes place, then it can be synchronous. So, they are called same semi synchronous or basically uses both synchronous and asynchronous technology.

(Refer Slide Time: 29:34)



So, these are basic principles by which data transfer takes place and there is another very important concept of interrupts, it turns out. Now, suppose that, there has to be see the CPU is the actually to all the time doing calculations, various kinds of calculations. And now, it has to do calculations as well as it has to do input output in the sense that, it can do can read signals.

Now, why it is doing and it is doing many calculations, so various kinds of calculations, so how does it know, when it needs to read some signal, whether the signal is there or not. So, there has to be a mechanism by which other devices can tell the CPU, which is

actually busily doing some calculation, now I there is some signal available here, which you need to read and they use in the near future possibly, but now the signal is present please read it.

Please leave your calculation for some time, come and read this signal and then you decide, whether to work on this signal or whether to go back to your calculation. So, basically this achieved by what is known as an interrupt, so while the CPU is doing something, interrupt is actually a mechanism by which external or internal events can change the course of programming execution.

So, the program execution is going on serially and then if an interrupt comes for some time, that program execution thread will be stopped and the CPU will do something else and then may be depending on the logic to an either come back to it or may start some other third execution. So, this in built into the CPU hardware, the CPU hardware is built in such a manner, that it automatically response to interrupts and such interrupts are called hardware interrupts.

Similarly, such things can also be done by sometimes by executing some software instructions, which are called software interrupts and then once interrupts arrive, this interrupts service routine, then there is a piece of program, which gets executed, that is called the in interrupt service routine, which handles the system interrupts. And then there are ways of various interrupts, you can understand that, while an interrupt service routine is going on another interrupt can also come.

So, one has to decide that which interrupts is actually more important and which interrupt is less important and things like that. So, these interrupts can also have their own priorities and accordingly they are response that is decided. But, since the system is all the time going through interrupt and syncing interrupt are such important things, that the CPU will instantly leave, what it is doing and at least take a look at, what interrupt is there it is so important.

Therefore, these interrupts are we support in responses, must be very quick and absolutely predictable, it should not happen that some interrupt is coming and that interrupt can possessing can take lot of time. Then, the system response will never remain predictable or bounded, so therefore, all care is taken, such that people can write

a interrupt service routines, which with bounded delays. So, interrupts are very highly use things in embedded systems.

(Refer Slide Time: 33:02)



Now, this after the hardware, so now we have got this hardware with various processors and with various input, output devices and these interrupt mechanisms and memory. So, now we have to use this piece of hardware and put programs in them, such that they can do some meaningful work. Now, obviously the software's will actually achieve, this various functionality, that is possible from embedded systems and typically embedded software has certain properties, like for example they have to be timely.

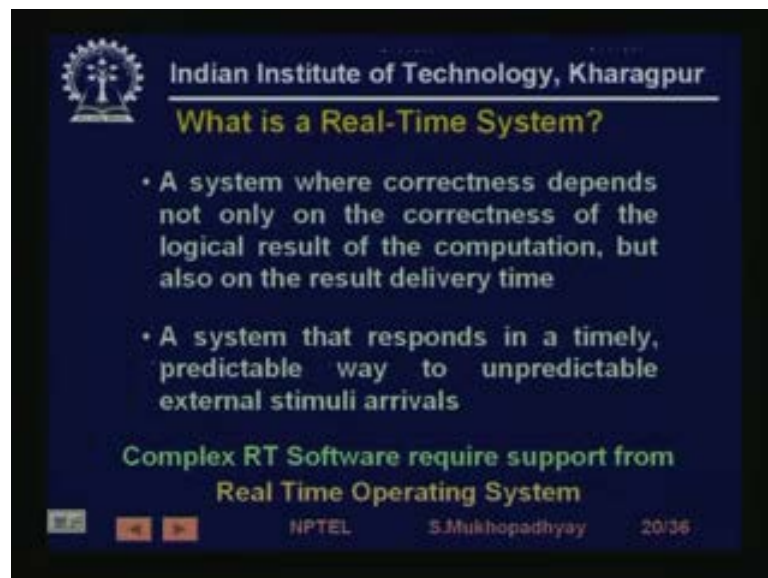
In the sense, that their executions must be related to times, that they must finish within certain times or them, they will they will occur with certain regularity of time. So, there they re they are related to time, that is why they are sometimes called real time. Then, they require concurrency in the sense that, there are several tasks all of which need to be processed together.

So, it is not that, where at least it should virtually be together, although strictly speaking it is going to be sequential, because there is only one processor, but it cannot happen that, so all of them must be processed together. We cannot because suppose one computer is monitoring five control loops, so naturally the programs of these things must occur. So, this control must go on concurrently by very fast time duration multiplexing sometimes this, sometimes this, sometimes this and then come back to this.

So, virtually you see think that all the works are being there and the liveness, I mean a situation should not occur, that everything stops in the computation always something should go on. Then, there are various, it must handle various kinds of interfaces to various kinds of devices, which are heterogeneous and they must be reactive to the environment, that is when certain conditions. So, mechanisms of interrupts etcetera must be present, I mean the software must handle these things.

So, these some properties which typically embedded system software must have and that is why, this software the some generally called real time systems. Because, they are generally concerned with time and they are generally real time systems are not necessarily, but generally real systems are reactive and concurrent.

(Refer Slide Time: 35:45)



Indian Institute of Technology, Kharagpur

What is a Real-Time System?

- A system where correctness depends not only on the correctness of the logical result of the computation, but also on the result delivery time
- A system that responds in a timely, predictable way to unpredictable external stimuli arrivals

Complex RT Software require support from Real Time Operating System

NPTEL S.Mukhopadhyay 20/36

So, what is the real time system, so real time system is system, where correctness, that is the functional correctness, that if you suppose give you a program, a program which let us say calculate, let us say that two control algorithm. So, one control algorithm may be very advanced and may be calculating very good control, but if cannot calculate within a sampling time, then that controller algorithm is not good.

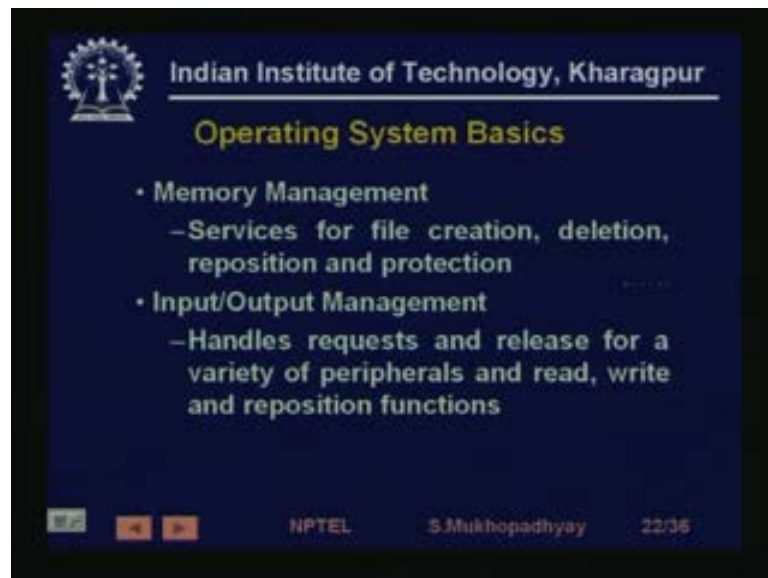
So, it is not correct that solution to the embedded control problem is not correct, that is of embedded control program, it may imply very sophisticated algorithm, but since it cannot compute it is value in time, it is not a correct solution. On the other hand, another

algorithm which may be approximate, but produces the solution in time is correct, so not only on the logical result, but also on the result delivery time.

So, a system that response in a timely and predictable way to unpredictable means external stimuli, so real time systems or embedded systems for industrial automation must be built. In such this system, it must have these properties, which is to be realized by through the software. Now, for realizing this software especially, when it has to do complex signal processing with a number of signals and a whole array of tasks, it becomes very difficult to achieve correctness.

It should not have any mistakes, so is for ease of development and for ease of running this kind of complex RT software, generally are run within an environment of an operating system. So, they generally is complex real time software, generally require support from a real time operating system, which actually basically provide some services, provide some tools, some handle, some small, small which this software can use the achieve, it is purpose.

(Refer Slide Time: 37:55)



So, now operating system is basically has four main tasks, one is process management, so there are at any time as I am saying that multiple task must be managed. So, the new task get dynamically created they have to be created, they had to be scheduled, that is one has to decide, when to execute which task, they have to be loaded on to the memory.

Finally, they while they are executing, one has to keep the control, they have to be terminated, remove from all sorts of things have to be done, so that all these related to that circle process management. Then, there has to be inter process communication, these tasks are actually not unrelated. So, they are quite related and they will exchange data, so the result produced by one has to be used by another.

So, all kinds of synchronization are necessary data exchange mechanism can necessary among tasks, so the operating system provides mechanisms by which application programs can actually do that. They provide process protection, so that one process does not corrupt the get of another process. Similarly, that provides memory management, so some task requires, how much memory it will require, what will be the address space all these things.

Similarly, input output management, so these are some of the crucial low level functions, which the operating system provides. So, using these features, one can write programs, which will use these features to quickly develop correct and efficient solutions.

(Refer Slide Time: 39:30)

Indian Institute of Technology, Kharagpur

What is a Real-time OS?

- An RTOS (Real-Time Operating System)
 - Is an Operating Systems with the necessary features to support a Real-Time System
- A Good RTOS is one that has a bounded (predictable) behavior under all system load scenarios
- Provides computational services. Does not guarantee system correctness

NPTEL S.Mukhopadhyay 23/36

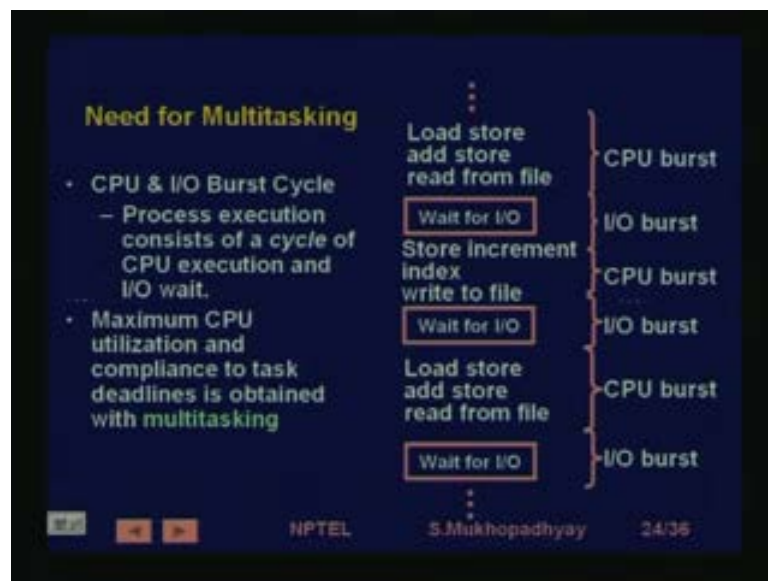
Apart from that, a real time system has some special features, which make which help to develop real time and be real time operating systems are required for developing real time systems. So, it will have some features such that, so bounded behavior can be realized is not that if you ask for another one kilo byte of memory, then to get that kilo

byte of memory, you can sometimes require 1 millisecond, sometimes require 50 milliseconds, you get this sort of thing or you try to access one particular variable.

So, sometimes you get that variable value within may be microsecond and sometimes you get it, you do not get in several milliseconds. So, if you have this kind of mechanisms, then your system response, there will be no predictable behavior, you cannot predict the behavior of your programs in a real situation. So, real time operating system insures that it is possible to built no predictable programs.

But, at the same time one must remember, that operating system mainly provides some computational service as such it cannot ensure that your application is right or it will work in the proper manner. That is to be designed by whoever is designing the system, but the operating system nearly provides an environment in which it is easier and possible to develop this.

(Refer Slide Time: 41:04)

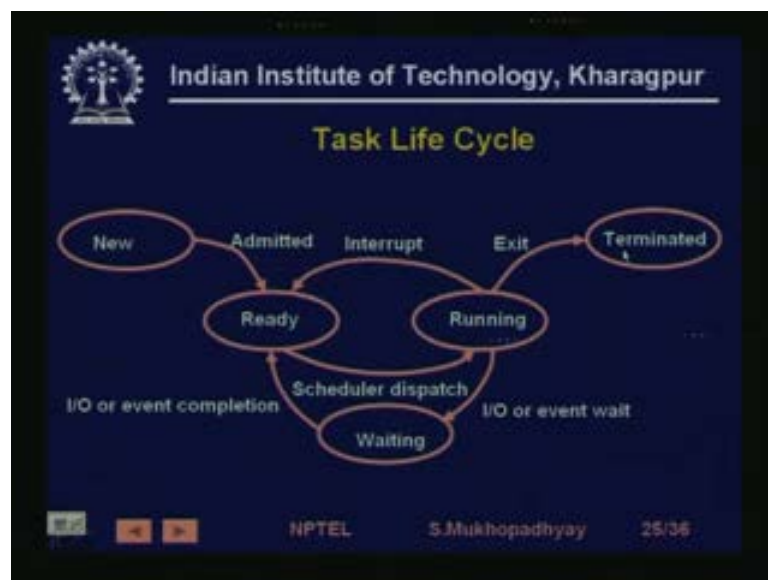


Now, real time operating systems necessarily have to multi tasked, because all the tasks, I mean every task has to be performed. First of all, tasks are sometimes repetitive and they have their own time deadlines and therefore, very good utilization of the CPU time needs to be done. So, you can see that here, what we are showing is that, typically computation is input output, it goes through a cycle of computation and input output.

Now, while input output is going on in many cases the CPU need not be busy, so it need not be idle also. So, therefore, it makes good sense to say that, if one task require some input output, then let it do the input output, while the CPU can use this time, because input output devices are also generally slower than the CPU.

So, by the time some 1 kilobyte of data will get transferred the CPU can do many much more computation. So, it should switch, it should keep the this task in stored, while it is waiting for IO and then go to another task and then do it is computation, that is better utilization of time. So, such a thing is called multi tasking and it is extremely important for real time systems to for the best utilization, so that good timing dead lines can be achieved.

(Refer Slide Time: 42:35)



So, under such a multi tasking scenario every task, in it is life cycle, it goes through several phases, like for example, when the task is first created, it is created, so it is called news. Then, it is admitted into that is ready to execute, some memory space has been allocated to it etcetera. Then, it is there put in some sort of a Q, where it is put, it is called ready, so now it is ready to be executed.

Now, some task is running and now while it is running, so many things can happen, either the task can go for an input output. So, if it goes for an input output, that running task is now put as a waiting state, if is removed and put into a waiting state, because it is now waiting for an input output or event. So, once it is waiting, now immediately there is

some part of the operating system will now decide that during this time while this task is waiting.

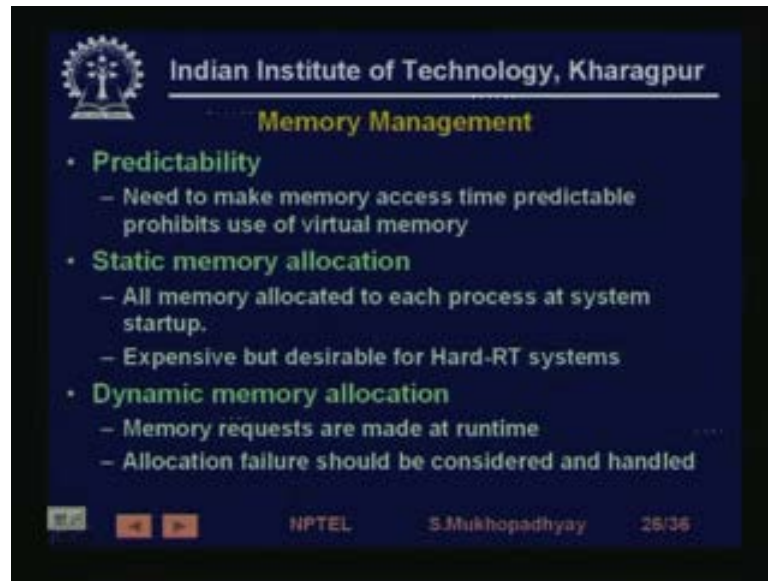
Where, which task will run, so some now among from the ready tasks, this scheduler or some part of the program, which is responsible for the scheduling will decide will take one of this task and make it ready. Now, while this task is waiting for, this old task is waiting for IO after some time the IO will get completed. So, when it gets completed, then it will move to the ready state, so it will be moved to the ready state and then it will be waiting.

So, on the other hand, one task may finally, get terminated, so once in get terminated, then again the scheduler will look at the ready Q and C, which one it has to schedule or there may be some interrupt coming. So, if the interrupt comes, then immediately the scheduler goes to see the interrupt and then see that, after it does some part of the interrupt processing.

It decides, now what to do, should it go back to the original running task or should it schedule, some new task or should it schedule do some computation related to the interrupting task. So, in other words, what are I am trying to say is that always a task goes through these kinds of states. So, it goes, it becomes ready, then it becomes, then it goes through this cycle, this ready, running, waiting, ready running waiting.

Or sometimes it executes, ready running, ready running cycles and then after through this, it is computation goes on. And then finally, it terminates itself. When terminates means, when the computation finally, finishes for that task, then it may exit. So, this a typical computational task like cycle in the embedded system.

(Refer Slide Time: 45:43)



Indian Institute of Technology, Kharagpur

Memory Management

- **Predictability**
 - Need to make memory access time predictable prohibits use of virtual memory
- **Static memory allocation**
 - All memory allocated to each process at system startup.
 - Expensive but desirable for Hard-RT systems
- **Dynamic memory allocation**
 - Memory requests are made at runtime
 - Allocation failure should be considered and handled

NPTEL S.Mukhopadhyay 25/35

Now, similarly this was about process management, then in a memory management, embedded systems have some typical things like for example, memory access time, such I said should be predictable. So, there is a mechanism of virtual memory, where because of shortages of RAM, people there are a strategy by which you have a concept of virtual memory.

So, as if an address is not found in that, if a particular variable is not found in the RAM, then there is a mechanism by which you can access the disc and then find it out. So, you can have a much since is the disc has the slow, but disc has the much larger memory space. So, you are the address space, that you can use in your program can be made very large, so this called of this called a virtual memory system.

But, embedded systems often prohibit is the use of virtual memory systems, because they can make it a memory accessible very memory access unpredictable. So, if it is in the RAM, it will be phased in micro seconds, if it is not in the RAM and if it is goes to a disc, which is electromechanical device, then it can take several milliseconds. So, for that the sometimes say that virtual memory should not be used, similarly static memory allocation, that is in the beginning programs, whole memory is allocated.

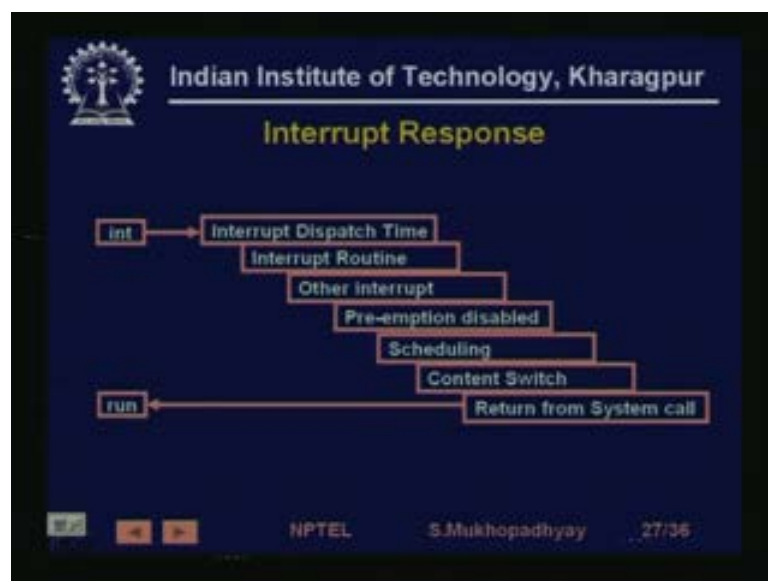
It is not that dynamic memory allocation means, whenever a program needs memory; it will be given some memory and when it does not memory need can be released also. Now, these strategies are in terms of memory utilization, they are good, static memory

allocation is actually expensive and at the same time, sometimes it may happen that, some peculiar situation may arise in which one task is actually asking of memory. But, it is no getting it, because there is no memory in the system.

So, if you have dynamic memory allocations, then such things can occur, right in the middle of execution of a task. If in static memory allocation, you already find out take worst case and then provide for enough memory. So, of course the system cost will increase, but for an embedded systems, since the control much bigger system, so such cost such generally not problems.

So, therefore, dynamic memory allocation is sometimes avoided in the case of embedded systems or even if you have it, then you must provide for situations. That, if there is a memory allocation failure, then what is going to happen, so you must be prepared for this strategies at run time. So, such features have there for memory management, similarly interrupt response.

(Refer Slide Time: 48:34)



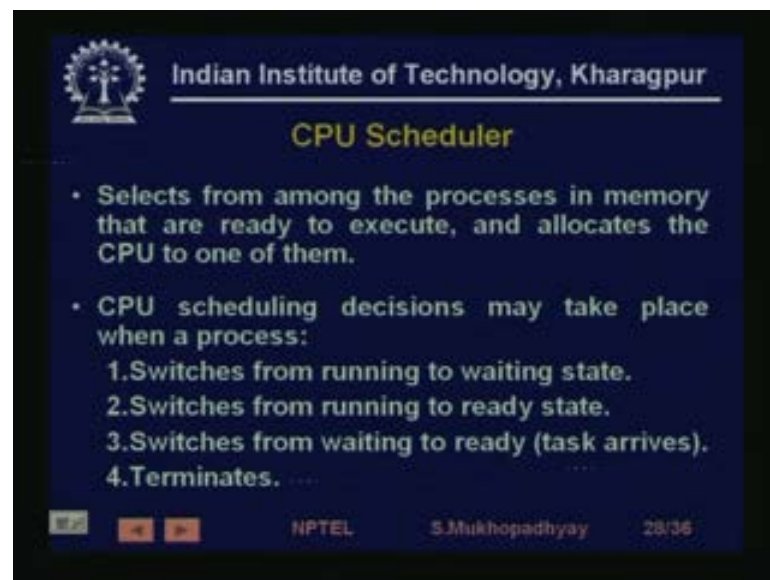
So, various systems have various interrupt response, so when an interrupt comes, first the interrupt is coming, then the interrupt service of routine is of control goes to the interrupt service are routine. Then, in the interrupt service of routine, one sees what are the other pending interrupts, which interrupts should be interrupt should be executed. In some times, some interrupts are so important, that temporarily what one has to do is that, it has to disable interrupt.

In the sense of there are some parts of the code, which are called critical sections, which must be executed continuously, so one must ensure that no other interrupt will come, while that small part of the code is getting executed. So, before that code execution preemption is disabled, that is control cannot be preempted, those tasks will serially be executed.

And then we know this critical section is executed, then some scheduling decisions are taken and then finally, after this scheduling a decision is taken as to which task is going to run now. So, then the content which means, one task was running, all its registers were actually loaded with the values, they have to be stored in the proper locations or stacks and then the next task, which has to be run, their values have to be loaded.

So, this called context switch, actually it is written as content switch, that is wrong is called a context switch and then finally, it will return from the system. So, all these things happen, every time an interrupt comes and what are the steps as we have seen of this interrupt responses is the scheduling decision.

(Refer Slide Time: 50:17)



Indian Institute of Technology, Kharagpur

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- CPU scheduling decisions may take place when a process:
 1. Switches from running to waiting state.
 2. Switches from running to ready state.
 3. Switches from waiting to ready (task arrives).
 4. Terminates. ...

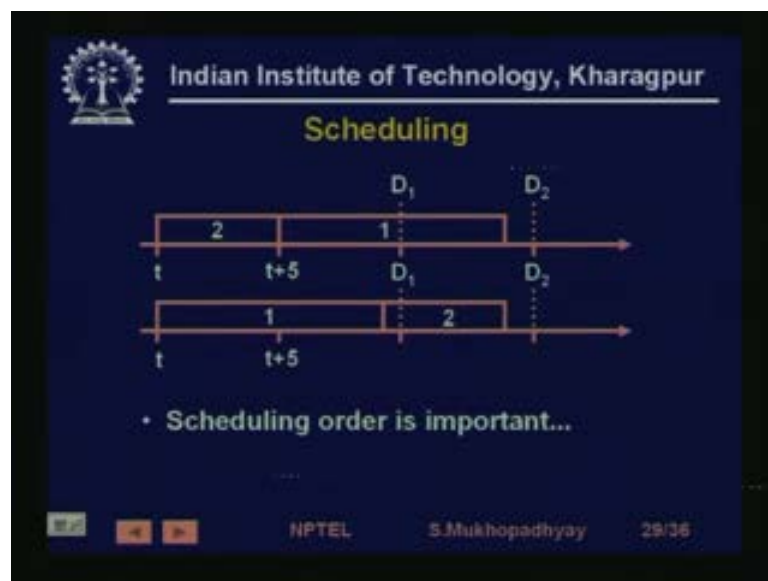
NPTEL S. Mukhopadhyay 28/36

So, now a every time this scheduler is invokes, scheduler is again a program, every time a scheduler is invoke, the scheduler decides that which task is to be run. So, as we have seen in during the life cycle, that CPU scheduling decisions may can be taken or continuously this scheduler exit examine. So, suppose a task switches from running to waiting state, it is waiting for some IO or some event.

So, then it will execute some software interrupt by which it will tell that I can wait now, I am waiting for IO and let the scheduler decide which task to run. So, during this time the scheduler can switch this task out, it may it can do a context switch and switch and put another task and then execute it while this task is waiting. Similarly, some such which is can happen, when it is going from running to ready state that is if another interrupt comes.

Every time an interrupt comes, the scheduler is invoked and the scheduler again takes a look at the ready tasks and their priorities and their importance, then decides which one is going to be run now. Similarly, if some such task goes from switching from waiting to ready, after an IO event completion, then also it can be various points of time. When, tasks are executed continuously periodically the scheduler takes a look at the ready task and then decides which task will be run now.

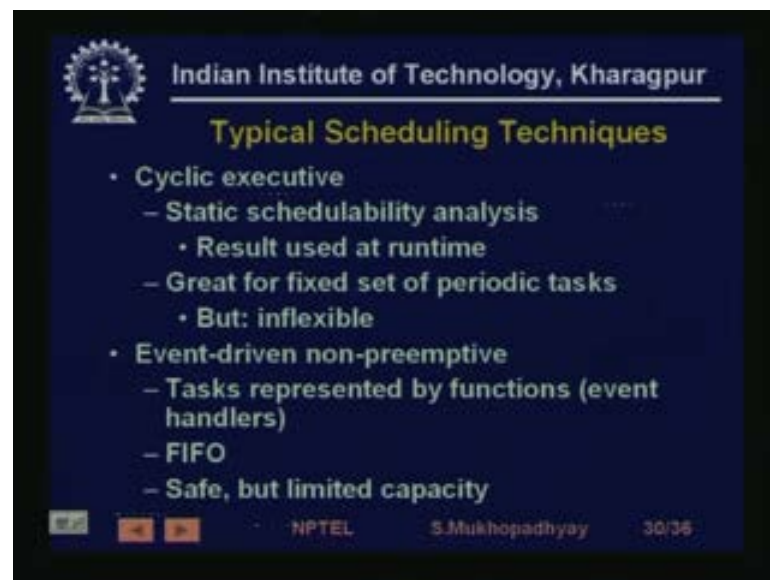
(Refer Slide Time: 51:51)



And for scheduling, there are see scheduling is very important, like for example, there are two tasks here, now task 2, takes 5 time unit is for execution, but it is deadline, that it should be completed after long time, this deadline for task 2. While, task one takes this much amount of time and it has a deadline of D_1 , so obviously, if you execute task 2 and then try to execute task 1. Then, the dead line of task one will be valid, so task 1, could not complete by it is deadline that is that is a fault.

On the other hand, so obviously this the trivial case; obviously you should execute task 1 first, so that it meets its deadline and then execute task 2, so that it also meets its deadline, so this is a correct solution. So, you see that scheduling is important, there is the order in which these tasks are going to be executed.

(Refer Slide Time: 52:46)



So, there are several scheduling techniques, which can be used, simplest of them is the cyclic executing, so this says that you have n number of tasks and simply go on executing them in cycle. So, first do task number 1, then 2, then 3, then 4, then 5 and come back to 1. So, cyclically execute this task, now this will be possible, when the number of tasks is fixed, when they can be cyclically executed, but there are several situations, when tasks can be generated in terms in response to events.

So, then we cannot execute them cyclically and we have to sometimes task will be generated, sometimes tasks will be terminated. So, for such even driven situations, you can have two kinds of scenarios, one kind of scheduling is called non preemptive scheduling, that you actually decide in what order to run to run these starts. But, once you have started one task you actually complete it and once it is completed, then you again decide which task to run.

So, you do not preempt a task, while it is being executed, that is called non preemptive, but as it turns out that that, again cannot give you the maximum amount of flexibility.

(Refer Slide Time: 54:06)

Indian Institute of Technology, Kharagpur

Scheduling

- Static/dynamic priority preemptive scheduling
 - Static schedulability analysis
 - No explicit schedule: at run-time execute highest priority task first
 - Rate monotonic, earliest deadline first,
 - Number of priority levels supported – 32 to be RT-POSIX compliant; many offer between 128-256
 - Type of scheduling for equal priority threads – FIFO or Round-Robin
 - Thread priorities be changed at run-time

NPTEL S.Mukhopadhyay 31/36

So, the strategy which is actually used for embedded systems scheduling is preemptive scheduling. So, in preemptive scheduling, every time, I mean the moment, you have a higher priority ready task, the execution of a lower priority task is going to be stopped and the higher priority task will start. So, the task execution can be preempted at any point of time. Now, this priority business, how you are going to assign priority, so you can have a static method or you can have a dynamic method, so if you have a dynamic method, so you can have two protocols.

(Refer Slide Time: 54:44)

Indian Institute of Technology, Kharagpur

Scheduling Protocol

- Rate monotonic with two tasks
- $C_1=2, D_1=5$
- $C_2=4, D_2=7$

J1 1 1 1 1 1

J2 2 2 2 2 2

0 2 4 6 8 10 12 14 16 18 20 22

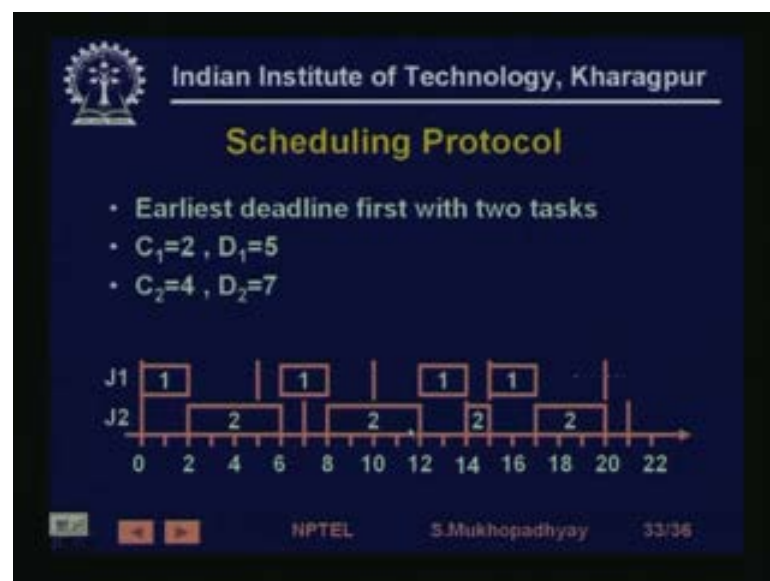
NPTEL S.Mukhopadhyay... 32/36

So, we have one protocol is rate monotonic, which says that task which have, rate monotonic protocol is typically applicable for periodic tasks. So, it says that, if a task has a lower rates at which it is going to be executed for example, here the deadline is 5. So, it is going to be called, once every 5 units of time, while D_2 is 7, so if going to be called every 7 units of time, so task 1 is actually more frequent.

So, therefore, it is more priority, it is more important, because it is more frequent, that is a strategy that you do for it monotonic scheduling. So, you see that, what is happens, suppose task 1 and task 2 are both required. So, task 1 is computed, since one is higher priority, it will become, it will be done. Then, 2 get executed and since 1 is not called, now here 1 is executed, see 2 has not been finished, 2 requires 4 unit is of time, that is computational time C_2 .

While, this 1, 2, 3, but then task 1 was again called, because now it is 5 unit is of time, so the second call of 1 has come therefore, 2 has been stopped, so in this case since the priority is fixed, one always has higher priority. So, every time, there is a need requirement from computing one two is being stopped, so this the execution cycle and we can see that, with these two tasks, we can actually meet that deadlines. So, this called rate monotonic scheduling, which is generally useful for a fix number of periodic tasks.

(Refer Slide Time: 56:30)

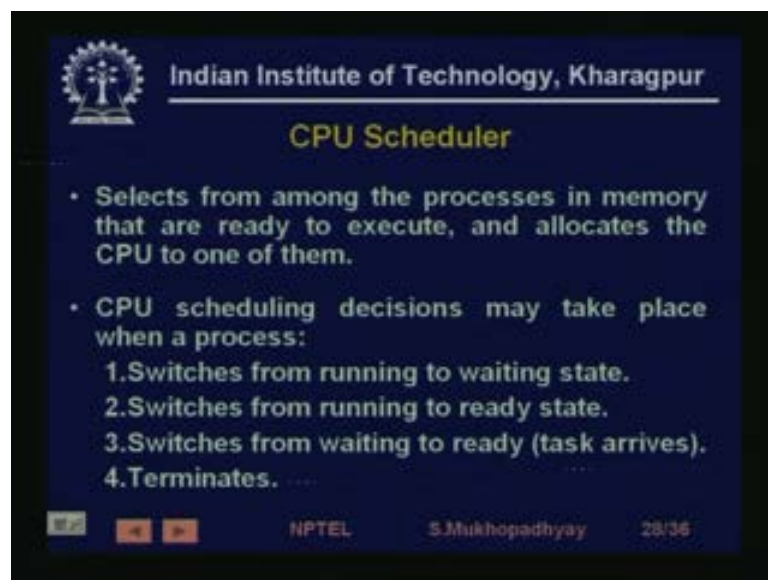


On the other hand, for greater CPU utilization, there is another strategy, which is also employed, which is called the earliest deadline first strategy, which says here the priority

is not fixed, here the priority can keep changing with time. So, whichever task has a closer deadline now has higher priority. So, write now 1 has a closer dead line, so 1 has higher priority.

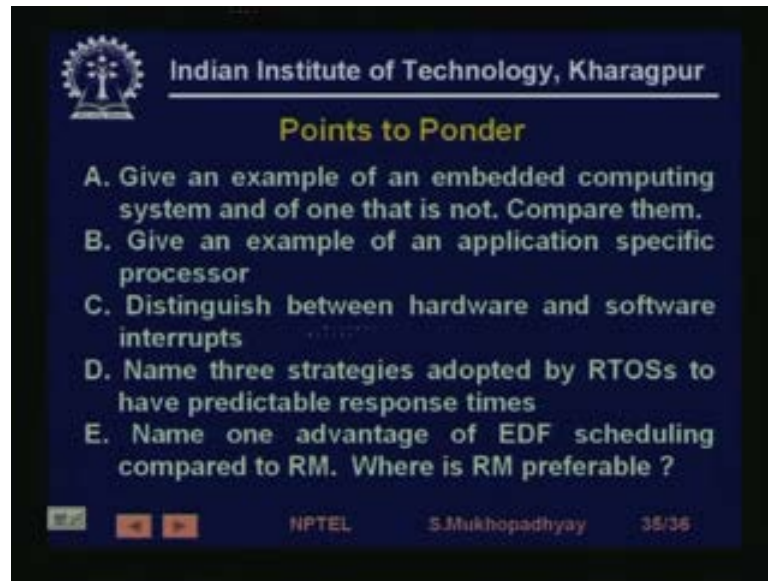
Now, after that 2 starts executing, now here you see that 2 will be again called at 5, but now the deadline of 2 is later than the deadline of 1 is later than this deadline of 2. So, therefore, 2 will continue to work, because at this time 2 have higher priority, because it is deadline is earlier. So, this difference and it can be shown, that you get better CPU utilization with this strategy, so this brings us to the end of this lesson.

(Refer Slide Time: 57:28)



So, we have seen is that, we have in this lesson, we have seen applications of embedded systems in industrial automation and you have seen the typical hardware components that are present, processors, memory, various kinds of IO. And we have seen the characteristics of embedded software that they have to be timely, they have to be concurrent, they have to be reactive. And we have seen that, generally you need for ensuring building systems, you need real time operating systems and we at the also took at real time task scheduling.

(Refer Slide Time: 58:07)



Indian Institute of Technology, Kharagpur

Points to Ponder

- A. Give an example of an embedded computing system and of one that is not. Compare them.
- B. Give an example of an application specific processor
- C. Distinguish between hardware and software interrupts
- D. Name three strategies adopted by RTOSs to have predictable response times
- E. Name one advantage of EDF scheduling compared to RM. Where is RM preferable ?

NPTEL S.Mukhopadhyay 35/36

Now, you can have several questions like an you can have your own example of an embedded computing system and find over one that is not. I gave give an example of an application specific processor, if you see the net you will find these. Distinguish between hardware and software interrupts, any micro processor book will give you this answer.

Name three strategy is adapted by RTOSs to have predictable response times, we have been mentioned in the lecture and name one advantage of EDF scheduling compared to rate monotonic. So, rate monotonic preferable for periodic tasks and they have EDF as better CPU utilization, so that is all for today.

Thank you very much.