Course Code	Course Name	Teaching Scheme			Credits Assigned				
		Theory	Practical	Tutorial	Theory	Practical	Tutorial	Total	
EXC701	Embedded	04			04			04	
	System Design								

Course	Course Name	Examination Scheme							
Code		Theory Marks				Term	Practical	Oral	Total
		Internal assessment End Sem.				Work			
		Test 1	Test 2	Ave. Of	Exam				
				Test 1 and					
				Test 2					
EXC701	Embedded	20	20	20	80	-	-	-	100
	System Design								

Course Pre-requisite:

• EXC403: Microprocessor and Peripherals

• EXC501: Microcontroller & Applications

Course Objectives:

- 1. To teach scope, usage, requirements, challenges and general design methodology of embedded system
- 2. To apply hardware and software knowledge to develop embedded system applications according to requirement and constraints

Course Outcomes:

After successful completion of the course student will be able to

- 1. interpret component's functional and electrical specifications and its implication and advantage in design.
- 2. develop their skill to select/choose proper components, approach, and method to develop optimal system.

1.1 Core of the embedded System No.	Module	Unit	Topics	Hrs.			
1.1 Core of the embedded system, Memory, Sensors (resistive, optical, position, thermal) and Actuators (solenoid valves, relay/switch, opto-couplers), Communication Interface, Embedded firmware (RTOS, Drivers, Application programs), Power-supply (Battery technology, Solar), PCB and Passive components, Safety and reliability, environmental issues. Ethical practice. 1.2 Characteristics and quality attributes (Design Metric) of embedded system. Real time system's requirements, real time issues, interrupt latency. 1.3 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UML. 2 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UML. 3 Embedded Product development life sple, Wireless sensor network 5 Embedded Hardware and Design 1.0 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM de motor control as a case study using on chip timers and watch-dog-timers). 3.1 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-M; (Cortex-M3). The programs of ARM-v7-A (CortexA8), ARM-v7-M; (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-M; (Cortex-M3). The programs of ARM-v7-A (CortexA8), ARM-v7-M; (Cortex-M3), and the programs of ARM-v7-A (CortexA8), ARM-v7-M; (Cortex-M3). The programs of ARM-v7-A (Cortex-M3). The programs of		No.	E de contact de CE de la la la Contact	0			
Actuators (solenoid valves, relay/switch, opto-couplers), Communication Interface, Embedded firmware (RTOS, Drivers, Application programs), Power-supply (Battery technology, Solar), PCB and Passive components, Safety and reliability, environmental issues, Ethical practice. 1.2 Characteristics and quality attributes (Design Metric) of embedded system. Real time system's requirements, real time issues, interrupt latency. 1.3 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UML 2.1 Embedded Serial Communication 2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), I ² C, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network 3. Embedded Hardware and Design 3.1 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM dc motor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-R-(CortexR4), ARM-v7-M (Cortex-M3)). 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology 4.1 Embedded Software, Firmware Concepts and Design 4.1 Embedded Software, Firmware Concepts and Design 4.2 Embedded Software, Firmware Concepts and Design 4.3 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Applicatio	1	1 1		8			
Embedded firmware (RTOS, Drivers, Application programs), Power-supply (Battery technology, Solar), PCB and Passive components, Safety and reliability, environmental issues. Ethical practice. 1.2 Characteristics and quality attributes (Design Metric) of embedded system. Real time system's requirements, real time issues, interrupt latency. 1.3 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UMI. 2 Embedded Serial Communication 2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), I*C, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network 3 Embedded Hardware and Design 3.1 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM demotor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-M (Cortex-M3). 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology 4 Embedded Software, Firmware Concepts and Design 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrain μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Ap		1.1					
technology, Solar), PCB and Passive components, Safety and reliability, environmental issues. Ethical practice. 1.2 Characteristics and quality attributes (Design Metric) of embedded system. Real time system's requirements, real time issues, interrupt latency. 1.3 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UML. 2.1 Embedded Serial Communication 2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), I*C, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network 2.1 Low power hardware and Design 3.1 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM dc motor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA4), ARM-v7-R (CortexA4), AR							
1.2 Characteristics and quality attributes (Design Metric) of embedded system. Real time system's requirements, real time issues, interrupt latency. 1.3 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UML 2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), Γ'C, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network 2.1 Embedded Hardware and Design 3.1 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM demotor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-R (CortexA9), ARM-v7-M (Cortex-M3)). 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphors, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II, Inter-ta							
1.2 Characteristics and quality attributes (Design Metric) of embedded system. Real time system's requirements, real time issues, interrupt latency. 1.3 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UMI. 2 Embedded Serial Communication 4 2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), I²C, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network 12 3.1 Low power hardware and Design 4 3.2 Introduction to ARM-v7-M (Cortex-M3) based Real time clock and PWM demotor control as a case study using on chip timers and watch-dog-timers). 12 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology 2 4 Embedded Software, Firmware Concepts and Design 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance , Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II, Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Introduction Linux OS, Linux IPC usage, ba							
System's requirements, real time issues, interrupt latency. 1.3 Embedded Product development life cycle, Program modeling concepts: DFG, FSM, Petri-net, UML 2 Embedded Serial Communication 2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), FC, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network 3 Embedded Hardware and Design 3.1 Low power hardware design (MSP430 / Cortex-M3) based Real time clock and PWM demotor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-R (CortexA94, ARM-v7-M (Cortex-M3)) 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology Embedded Software, Firmware Concepts and Design 16 4.1 Embedded C-programsing concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II, Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and To		1.2	*				
Petri-net, UML Embedded Serial Communication 2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), I²C, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network Embedded Hardware and Design 3.1 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM dc motor control as a case study using on chip timers and watch-dog-timers). 12 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), Arm-v7-A (Co							
2.1 Study of basic communication protocols like SPI, SCI (RS232, RS485), I ² C, CAN, Fieldbus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network 3.1 Low power hardware and Design 3.1 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM dc motor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-R (CortexA4), ARM-v7-M (Cortex-M3). Comparison of ARM-v7-A (CortexA8), ARM-v7-R (CortexA9), ARM-v7-M (Cortex-M3). 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology Embedded Software, Firmware Concepts and Design 6 Fine Medded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II, GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware buesigning Methodology and Tools 6.1 Embedded System Designi		1.3					
Study of basic communication protocols like SPI, SCI (RS232, RS485), I²C, CAN, Field-bus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network			Petri-net, UML				
Sus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network	2			4			
Sus (Profibus), USB (v2.0), Bluetooth, Zig-Bee, Wireless sensor network		2.1	Study of basic communication protocols like SPI, SCI (RS232, RS485), I ² C, CAN, Field-				
 3.1 Low power hardware design (MSP430 / Cortex-M3 based Real time clock and PWM demotor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-R (CortexA8), ARM-v7-M (Cortex-M3) 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology 4.1 Embedded Software, Firmware Concepts and Design 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance , Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 6 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6 Requirement analysis, Hardware blocks di							
motor control as a case study using on chip timers and watch-dog-timers). 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-R (CortexR4), ARM-v7-M (Cortex-M3). 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology Embedded Software, Firmware Concepts and Design 16 4.1 Embedded Software, Firmware Concepts and Design 16 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools Simulation, Testing and Debugging Methodology and Tools Simulation, Testing and Debugging Methodology and Tools Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated Control and control and control and control and control and control and contro	3		Embedded Hardware and Design	12			
 3.2 Introduction to ARM-v7-M (Cortex-M3), Comparison of ARM-v7-A (CortexA8), ARM-v7-R (CortexA4), ARM-v7-M (Cortex-M3) 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology Embedded Software, Firmware Concepts and Design 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 6 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical:		3.1					
 v7-R (CortexR4), ARM-v7-M (Cortex-M3) 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology Embedded Software, Firmware Concepts and Design 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 							
 3.3 Direct digital solution using CPLD, FPGA, its advantages, and introduction to related development methodology Embedded Software, Firmware Concepts and Design 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, proc		3.2	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \				
development methodology Embedded Software, Firmware Concepts and Design 16							
 4 Embedded Software, Firmware Concepts and Design 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 		3.3					
 4.1 Embedded C-programming concepts (from embedded system point of view): Optimizing for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. Simulation, Testing and Debugging Methodology and Tools GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 				4.6			
for Speed/Memory needs, Interrupt service routines, macros, functions, modifiers, data types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated	4	4.4		16			
types, device drivers, Multithreading programming. (Laboratory work on J2ME Java mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated		4.1					
 mobile application). 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 							
 4.2 Basic embedded C programs/applications for ARM-v7, using ARM-GCC-tool-chain, Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 							
Emulation of ARM-v7 (e.g. using QEMU), and Linux porting on ARM-v7 (emulation) board 4.3 Real time operating system: POSIX Compliance, Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated		12					
 board 4.3 Real time operating system: POSIX Compliance , Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. Simulation, Testing and Debugging Methodology and Tools GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. Embedded System Designing Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 		4.2					
 4.3 Real time operating system: POSIX Compliance , Need of RTOS in Embedded system software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 							
software, Foreground/Background systems, multitasking, context switching, IPC, Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated		4.3					
Scheduler policies, Architecture of kernel, task scheduler, ISR, Semaphores, mailbox, message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated							
 message queues, pipes, events, timers, memory management, RTOS services in contrast with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. Simulation, Testing and Debugging Methodology and Tools GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. Embedded System Designing Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 							
 with traditional OS. 4.4 Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. Simulation, Testing and Debugging Methodology and Tools GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. Embedded System Designing Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 			message queues, pipes, events, timers, memory management, RTOS services in contrast				
 μCOS-II, Inter-task communication in μCOS-II, Memory management in μCOS-II, porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. Simulation, Testing and Debugging Methodology and Tools GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. Embedded System Designing Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 							
porting of RTOS on ARM-v7 (emulation) board, Application developments using μCOS-II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 6 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated		4.4	Introduction to μCOS-II RTOS, study of kernel structure of μCOS-II, Synchronization in				
II. 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 6 SNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated							
 4.5 Introduction Linux OS, Linux IPC usage, basic device (drivers) usage. 5 Simulation, Testing and Debugging Methodology and Tools 6 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated 							
5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated			·				
5.1 GNU Debugger (gdb), Boundary-Scan/JTAG interface concepts, Black-box, White-box testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated		4.5		0.1			
testing, Hardware emulation, logic analyzer. 6 Embedded System Designing 08 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated	5		, c cc c st	04			
6 Embedded System Designing 08 6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated		5.1					
6.1 Requirement analysis, Hardware blocks diagram, System model (like FSM, UML), Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated	(00			
Software architectures (modules, drivers), and Component/hardware selection, covering following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated	0	<i>L</i> 1		U8			
following cases: Hard real time/ Mission critical: Missile, Car cruise control, medical monitoring systems, process control system (temp, pressure) Soft real time: Automated		0.1					
monitoring systems, process control system (temp, pressure) Soft real time: Automated			l · · · · · · · · · · · · · · · · · · ·				
routers, Wireless (sensor) networks.							
Total 52				52			

Recommended Books:

- 1. Embedded Systems, Rajkamal, TMH, 2008.
- 2. Frank Vahid Embedded Systems, Wiley India, 2002
- 3. ARM System-on-Chip Architecture, Steve Furber Pearson 2005
- 4. Jean J Labrose MicroC / OS-II, Indian Low Price Edition 2002
- 5. DR.K.V.K.K. Prasad Embedded / real time system, Dreamtech
- 6. Iyer, Gupta Embedded real systems Programming, TMH
- 7. Embedded systems software primer, David Simon Pearson
- 8. ARM System Developers Guide- Sloss, Symes, Wright, ElsevierMorgan Kaufman, 2005
- 9. LPC2148 Data Sheets www.arm.com
- 10. ARM Programers/architectural manual.
- 11. MSP430 architectural manual.
- 12. Embedded Microcomputer Systems Real Time Interfacing Jonathan W. Valvano; Cengage Learning; Third or later edition.

Internal Assessment (IA):

Two tests must be conducted which should cover at least 80% of syllabus. The average marks of both the test will be considered for final internal assessment.

End Semester Examination:

- 1. Question paper will comprise of 6 questions, each of 20 marks.
- 2. Total 4 questions need to be solved.
- 3. Question No.1 will be compulsory and based on entire syllabus wherein sub questions of 2 to 5 marks will be asked.
- 4. Remaining question will be selected from all the modules.