PANDIAN SARASWATHI YADAV ENGINEERING COLLEGE, ARASANOOR
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY – CHENNAI.

REGULATION 2008

B.E. – ECE
(IV Year / VIII SEM)

EC2042 - EMBEDDED AND REAL TIME SYSTEMS

QUESTION BANK

EC2042 - EMBEDDED AND REAL TIME SYSTEMS

L	T	P	C
3	0	0	3

UNIT I INTRODUCTION TO EMBEDDED COMPUTING **9**

Complex systems and microprocessors – Design example: Model train controller – Embedded system design process – Formalism for system design – Instruction sets Preliminaries – ARM Processor – CPU: Programming input and output – Supervisor mode, exception and traps – Coprocessor – Memory system mechanism – CPU performance – CPU power consumption.

UNIT II COMPUTING PLATFORM AND DESIGN ANALYSIS **9**

CPU buses – Memory devices – I/O devices – Component interfacing – Design with microprocessors – Development and Debugging – Program design – Model of programs – Assembly and Linking – Basic compilation techniques – Analysis and optimization of execution time, power, energy, program size – Program validation and testing.

UNIT III PROCESS AND OPERATING SYSTEMS **9**

Multiple tasks and multi processes – Processes – Context Switching – Operating Systems –Scheduling policies - Multiprocessor – Inter Process Communication mechanisms – Evaluating operating system performance – Power optimization strategies for processes.

UNIT IV HARDWARE ACCELERATES & NETWORKS **9**

Accelerators – Accelerated system design – Distributed Embedded Architecture – Networks for Embedded Systems – Network based design – Internet enabled systems.

UNIT V CASE STUDY **9**

Hardware and software co-design - Data Compressor - Software Modem – Personal Digital Assistants – Set–Top–Box. – System-on-Silicon – FOSS Tools for embedded system development.

TOTAL= 45 PERIODS

TEXT BOOK:

1) Wayne Wolf, “Computers as Components - Principles of Embedded Computer System Design”, Morgan Kaufmann Publisher, 2006.

REFERENCES:

1) David E-Simon, “An Embedded Software Primer”, Pearson Education, 2007.

2) K.V.K.K.Prasad, “Embedded Real-Time Systems: Concepts, Design & Programming”, dreamtech press, 2005.

3) Tim Wilmshurst, “An Introduction to the Design of Small Scale Embedded Systems”, Pal grave Publisher, 2004.

4) Sriram V Iyer, Pankaj Gupta, “Embedded Real Time Systems Programming”, Tata Mc-Graw Hill, 2004.

5) Tammy Noergaard, “Embedded Systems Architecture”, Elsevier,2006.

UNIT-I
INTRODUCTION TO EMBEDDED COMPUTING
2 MARKS

1. Define Embedded System. What are the components of embedded system?

An Embedded system is one that has computer hardware with software embedded in it as one of its most important component. The three main components of an embedded system are

1. Hardware
2. Main application software
3. RTOS

2. In what ways CISC and RISC processors differ?

CISC	RISC
It provides number of addressing modes	It provides very few addressing modes
It has a micro programmed unit with a control memory	It has a hardwired unit without a control memory
An easy compiler design	Complex compiler design
Provide precise and intensive calculations slower than a RISC	Provide precise and intensive calculations faster than a CISC

3. Define system on chip (SOC) with an example

Embedded systems are being designed on a single silicon chip called system on chip.SOC is a new design innovation for embedded system

Eg. Mobile phone.

4. Give any two uses of VLSI designed circuits

A VLSI chip can embed IPs for the specific application besides the ASIP or a GPP core. A system on a VLSI chip that has all of needed analog as well as digital circuits.

Eg. Mobile phone.

5. List the important considerations when selecting a processor.

Instruction set, Maximum bits in an operand, Clock frequency, Processor ability

6. What are the types of embedded system?

- Small scale embedded systems
- Medium scale embedded systems
- Sophisticated embedded systems

7. Classify the processors in embedded system?

1. General purpose processor
   - Microprocessor
   - Microcontroller

Embedded processor
Digital signal processor
Media processor

2. Application specific system processor
3. Multiprocessor system using GPP and ASSP GPP core or ASIP core integrated into either an ASIC or a VLSI circuit or an FPGA core integrated with processor unit in a VLSI chip.

8. What are the important embedded processor chips?

- ARM 7 and ARM 9
- i 960
- AMD 29050

9. Name some DSP used in embedded systems?

- TMS320Cxx
- SHARC
- 5600xx

10. Name some of the hardware parts of embedded systems?

- Power source
- Clock oscillator circuit
- Timers
- Memory units
- DAC and ADC
- LCD and LED displays
- Keyboard/Keypad

11. What are the various types of memory in embedded systems?

 RAM (internal External)
 ROM/PROM/EEPROM/Flash
 Cache memory

12. What are the points to be considered while connecting power supply rails with embedded system?

 A processor may have more than two pins of Vdd and Vss
 Supply should separately power the external I/O driving ports, timers, and clock and
 From the supply there should be separate interconnections for pairs of Vdd and Vss pins analog ground analog reference and analog input voltage lines.

13. What is watch dog timer?

Watch dog timer is a timing device that resets after a predefined timeout.

14. What are the two essential units of a processor in embedded system?

Program Flow control Unit and Execution Unit

15. What does the execution unit of a processor in an embedded system do?

The EU includes the ALU and also the circuits that execute instructions for a program control task. The EU has circuits that implement the instructions pertaining to data transfer operations and data conversion from one form to another.

16. Give examples for general purpose processor.

- Microcontroller
- Microprocessor

17. Define microprocessor.

A microprocessor is a single VLSI chip that has a CPU and may also have some other units for example floating point processing arithmetic unit pipelining and super scaling units for faster processing of instruction.

18. When is Application Specific System processors (ASSPs) used in an embedded system?

An ASSP is used as an additional processing unit for running the application specific tasks in place of processing using embedded software.

19. Define ROM image.

Final stage software is also called as ROM image .The final implement able software for a product embeds in the ROM as an image at a frame. Bytes at each address must be defined for creating the image.

20. Define device driver.

A device driver is software for controlling, receiving and sending byte or a stream of bytes from or to a device.

21. Name some of the software's used for the detailed designing of an embedded system.

- Final machine implement able software for a product
- Assembly language
- High level language
- Machine codes
- Software for device drivers and device management.

22. What are the various models used in the design of an embedded system?

- Finite state machine
- Petri net
- Control and dataflow graph
- Activity diagram based UML model
- Synchronous data flow graph
- Timed Petri net and extended predicate/transition net
- Multithreaded graph

23. Give some examples for small scale embedded systems.

- ACVM
- Stepper motor controllers for a robotic system
- Washing or cooking system
- Multitasking toys

24. Give some examples for medium scale embedded systems.

- Router, a hub and a gateway
- Entertainment systems
- Banking systems
- Signal tracking systems

25. Give some examples for sophisticated embedded systems.

- Embedded system for wireless LAN
- Embedded systems for real time video
- Security products
- ES for space lifeboat.

26. What are the requirements of embedded system?

- Reliability
- Low power consumption
- Cost effectiveness
- Efficient use of processing power

27. Give the characteristics of embedded system?

a) Single-functioned
b) Tightly constrained
c) Reactive and real time

28. What are the design metrics?

- Power
- Size
- NRE cost
- Performance

29. What are the challenges of embedded systems?

- Hardware needed
- Meeting the deadlines
- Minimizing the power consumption
- Design for upgradeability

30. Give the steps in embedded system design?

- Requirements
- Specifications
- Architecture
- Components
- System integration

31. What are the requirements?

Before designing a system, it must to understand what has to be designed. This can be known from the starting steps of a design process.

32. Give the types of requirements?

Functional requirements
Non-functional requirements

33. Define functional requirements.

It says the fundamental functions of an embedded system.

34. Give some examples of functional requirements.

1. Performance
2. Cost
3. Physical size and weight
4. Power

35. What is the use of requirements form?

It is used as a checklist in the requirements analysis. From this the fundamental properties of a system came to be known.

36. What are the entries of a requirement form?

- Name
- Purpose
- Inputs and outputs
- Functions
- Performance
- Manufacturing cost
- Power
- Physical size and weight

37. What is meant by specification?

This is a bridge between Customer and Architect. It conveys the customer's needs. These needs are properly used in the design process.

38. What is architecture design?

It says the way of implementing functions by a system. Actually architecture is a plan for whole structure of a system. While will bring the design of components later.

39. Define system integration

It is a processor of combining the components into one system.

40. What are the functions of memory?

The memory functions are

- To provide storage for the software that it will run.
- To store program variables and the intermediate results
- Used for storage of information

41. Define RAM.

RAM refers Random Access Memory. It is a memory location that can be accessed without touching the other locations.

42. What is data memory?

When the program is executing, to save the variable and program stack, this type of memory is used

43. What is code memory?

The program code can be stored by using this area. The ROM is used for this purpose.

44. What are the uses of timers?

The time intervals can be completed
Precise hardware delays can be calculated
The timeout facilities are generated

45. Give short notes on ARM processor.

It is said to be the family of RISC architecture. The ARM instructions are written one per line, starting after the first column.

46. What are the data types supported by ARM?

Standard ARM word is 32 bit long and Word is splitted into 4 8 bit bytes

47. What are the 3 types of operating modes?

- Normal mode
- Idle mode
- Power down mode

16 MARKS

1. List the hardware units that must be present in the embedded systems.

Power Source and managing the power

Most systems have a power supply of own. The supply has a specific operation range or a range of voltages. Various units in unzip for place of Infatuation to Embedded System in one of the following four accelerator are examples of embedded systems that do not have their own power supply v and connect to pi power-supply lines. 1Z; A charge pump consists of a diode in the series followed by a charging capacitor. The diode gets forward bias input from an external signal.

Real Time Clock (RTC) and Timers

Timing and Counting Needs of the System a timer circuit suitably configured is the system-clock, also called real-time clock (RTC). It is used by the schedulers and for real-time programming. More than one timer using the system clock (RTC) may be needed for the various timing and counting needs in a system.

Clock Oscillator Circuit and Clocking Unit(s)

The Clock is an important unit of a system. A processor needs a clock oscillator circuit. The clock controls the various clocking requirements of the CPU, of the system timers and the CPU machine cycles

Memories

In a system, there are various types of memories. They are as follows:

- Internal RAI\4 of 256 or 5lZ bytes in a microcontroller for registers,
- temporary data stack
- Internal ROM/PROMIEPROM for about 4 KB to 16 KB of program
- External RAM for the temporary data and stack
- Internal caches
- EEPROM or flash
- External ROM or PROM for embedding software
- RAM Memory buffers
- Caches (in superscalar microprocessors)

Pulse Dialer, Modem and Transceiver

- For user connectivity through the telephone line, wireless, or a system provides the necessary interfacing circuit.
- It also provides the software for pulse dialing through the telephone line, for modem interconnection for fax, for Internet packets routing, and for transmitting and connecting a WAG (Wireless Gateway) or cellular system.
- A transceiver is a circuit that can transmit as well as receive byte streams

Linking and Interfacing Buses and Units of the Embedded System Hardware

- The buses and units in the embedded system hardware are needed to be linked and interfaced.
- One way to do this is to incorporate a glue logic circuit.

LCD and LED Displays

- System requires an interfacing circuit and software to display the status or message for a fine, for multi-line displays, or flashing displays
- An LCD screen may show up a multilane display of characters or also show a graph or icon
- An LCD needs little power. It is powered by a supply or battery (a solar panel in the calculator). LCD is a diode that absorbs or emits right on appreciation of 3 V to 4 V and 50 or 60 Hz voltage pulses with currents less than 50 pA. An LSI (Lower Scale Integrated circuit) display controller is often used in the cise of matrix displays.

2. Explain the Exemplary applications of each type of embedded system.

Small Scale Embedded system

A long needle rotates elderly minute such that it 'returns to same position after an hour. A short needle rotates every hour. Such that it returns to same position after twelve hours

- ACVM
- Stepper motor controllers for a robotic system
- Washing or cooking system
- Multitasking toys

Medium scale embedded systems

- Router, a hub and a gateway
- Entertainment systems
- Banking systems
- Signal tracking systems

Sophisticated embedded systems

- Embedded system for wireless LAN
- Embedded systems for real time video
- Security products
- ES for space lifeboat.

3. Explain the various form of memories present in a system.

- RAM(internal External)
- ROM
- PROM
- EEPROM
- Flash
- Cache memory
- EEPROM or flash
- Extemal ROM or PROM for embedding software
- RAM Memory buffers
- Caches (in superscalar microprocessors)

4. Explain the software tools in designing of an embedded system.

Tools

- Editor
- Interpreter
- Compiler
- Assembler
- Cross assembler
- Simulator
- Source code engg software
- RTOS
- Stethoscope
- Trace Scope
- IDE
- Prototype
- Locator

5. Explain the processors in an Embedded System.

Processors in an ES

- General purpose processor
  - Microprocessor
  - Microcontroller
  - Embedded processor
  - Digital signal processor
  - Media processor
- Application specific system processor
- Multiprocessor system using GPP and ASSP
- GPP core or ASIP core integrated into either an ASIC or a VLSI circuit or an FPGA core integrated with processor unit in a VLSI chip.

Explanation

A processor has two essential units

- Program Flow Control Unit (CU)
- Execution Unit (EU)

An embedded system processor chip or core can be one of the following.

a. Microprocessor.
b. Microcontroller.
c. Embedded Processor.
d. Digital Signal Processor (DSP).
e. Media Processor.

6. What are the Challenges in Embedded systems?

How much hardware do we need?

- For the great deal of control over the amount of computing power, we cannot only select microprocessor used but also the amount of memory and peripheral device etc.
- The choice of hardware must meet both performance deadlines and manufacturing cost constraints.

How do we meet deadlines?

 Brute-force way to meet deadline by speedup the hardware so that the program runs faster.

 Increase in speed makes the system more expensive and also increasing the CPU clock rate.

How do we minimize power consumption?

 In battery powered applications, power consumption is very important.

 In non battery powered applications, excessive power consumption can increase heat.One way to consume less power is to run the system more slowly.

 But slow down the system can obviously lead to missed deadlines.

 Careful design is required to slow down the non critical parts of the machine.

How do we design for upgradeability?

 Hardware platform may be used over several product generations or for several different versions of a product in the same generation with few or no changes.

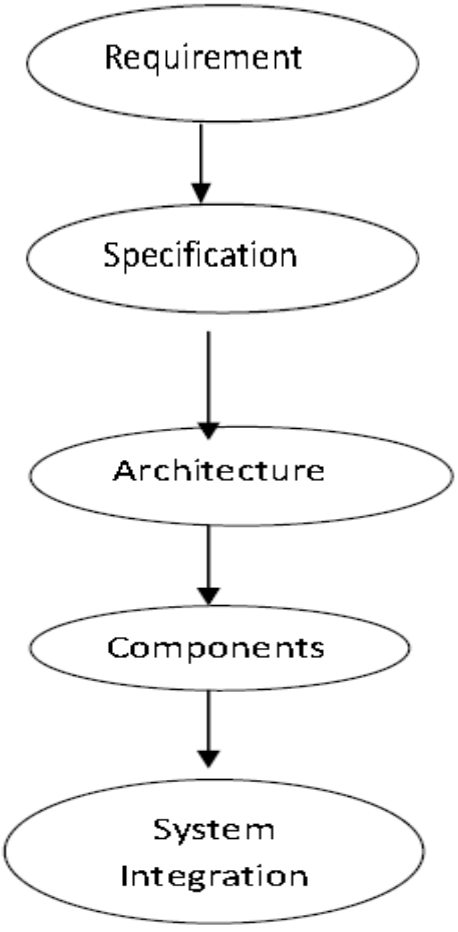
 Hardware is designed such that the features are added by changing software.

Does it really work?

 Reliability is very important when selling products.

 Reliability is important because running system and try to eliminate bugs will too late and fixing bugs will more expensive.

7. Give details about embedded system design process.



Requirements:

 We must know what we are designing, the initial state of the design process capture this information for use in creating the architecture and components.

 We gather an informal description from the customer known as requirements and we refine the requirement into specification that contain enough information to design the system architecture.

Requirement may be functional or non functional requirements include

Performance:

 The speed of the system is major consideration for the usability of the system and for its ultimate cost.

 Performance may be a combination of soft performance metric and deadline by which a particular operation must be completed.

Cost:

 Manufacturing cost

 Non-Recurring Engineering

Physical size and Weight:

 An industrial control system for an assembly line may be designed to fit into a standard size with no limitation and weight.

 A handheld device typically has tight requirement on both size and weight that can move to the entire system design.

Power consumption:

Power can be specified in requirement states in terms of

Battery Life

Describe the allowable voltage

Validating Requirements:

 Checking that a system meets specifications and fulfill its purpose.

 One good way to refine the user interface portion of a system requirement is to get a mock-up.

Simple Requirement Form:

Name:
Purpose:
Input:
Output:
Functions:
Performance:
Manufacture cost:
Power:
Physical size and weight:

Specification:

 It serves as the contract between the customer and the architecture.
 These specifications must be carefully written so that it reflects the customer's requirements and it helps during design.
 Designers who lack a clear idea what to build undergo a faculty assumption early in the process.
 Specification must be understandable
 Unclear specification will cause different types of problems.

Example for specification: GPS system

Architecture Design:

 The architecture is a plan for the overall structure of the system.
 It will be used later to design the component that makeup the architecture.

System Integration:

 Once the components are build up then they are integrated together and see the working system.
 Bugs are determined during integration.
 Careful attention to inserting appropriate debugging facilities during design can help ease system integration.

8. Explain about embedded system for digital camera.

Embedded system:

 An embedded system is a computer system design to perform a particular task or few dedicated functions. Eg: digital camera, calculator, cell phone
 It may be either an independent system or a part of a large system.
 Embedded system are controlled by one or more processing coder typically either microcontroller or DSP.
 It has application software and real time operating system (RTOS) in program memory and may perform series of task or multiple tasks.

Digital Camera:

 The charge couple Device (CCD) contains an array of light sensitive photocells that capture an image.
 The memory controller controls access to a memory chip also found in the camera, while the DMA controller enables direct memory access by other devices while the microcontroller is performing other functions.
 The LCD control and Display control circuits control the display of images on the camera's liquid crystal display device.
 The system always acts as a digital camera wherein it captures, compresses, and stores frames, decompresses and displays frames and upload frames.
 It is tightly constrained. The system must be lower cost so that the consumers able to afford such camera.
 It must be small so that it fits into within a standard sized camera.
 It must be fast so that it can process numerous images in milliseconds.
 It must consume less power so that the camera's battery will last long time.

UNIT-II
COMPUTING PLATFORM AND DESIGN ANALYSIS
2 MARKS

1. Differentiate synchronous and iso-synchronous communication.

Synchronous communication - When a byte or a frame of the data is received or transmitted at constant time intervals with uniform phase difference, the communication is called synchronous communication.

Iso-synchronous communication - Iso-synchronous communication is a special case when the maximum time interval can be varied.

2. What are the two characteristics of synchronous communication?

Bytes maintain a constant phase difference

The clock is not always implicit to the synchronous data receiver.

3. What are the three ways of communication for a device?

- Iso-synchronous communication
- synchronous communication
- Asynchronous communication

4. Expand a) SPI b) SCI

SPI—serial Peripheral Interface

SCI—Serial Communication Interface

5. Define software timer.

This is software that executes and increases or decreases a count variable on an interrupt from a timer output or form a real time clock interrupt. A software timer can also generate interrupt on overflow of count value or on finishing value of the count variable.

6. What is I2C?

I2C is a serial bus for interconnecting ICs .It has a start bit and a stop bit like an UART. It has seven fields for start,7 bit address, defining a read or a write, defining byte as acknowledging byte, data byte, NACK and end.

7. What are the bits in I2C corresponding to?

It has seven fields for start,7 bit address, defining a read or a write, defining byte as acknowledging byte, data byte, NACK and end

8. What is a CAN bus? Where is it used?

CAN is a serial bus for interconnecting a central Control network. It is mostly used in automobiles. It has fields for bus arbitration bits, control bits for address and data length data bits, CRC check bits, acknowledgement bits and ending bits.

9. What is USB? Where is it used?

USB is a serial bus for interconnecting a system. It attaches and detaches a device from the network. It uses a root hub. Nodes containing the devices can be organized like a tree structure. It is mostly used in networking the IO devices like scanner in a computer system.

10. What are the features of the USB protocol?

A device can be attached, configured and used, reset, reconfigured and used, share the bandwidth with other devices, detached and reattached.

11. Explain briefly about PCI and PCI/X buses.

PCI and PCI/X buses are independent from the IBM architecture .PCI/X is an extension of PCI and support 64/100 MHZ transfers. Lately, new versions have been introduced for the PCI bus architecture.

12. Why are SPCI parallel buses important?

SPCI serial buses are important for distributed devices. The latest high speed sophisticated systems use new sophisticated buses.

13. What is meant by UART?

UART stands for universal Asynchronous Receiver/Transmitter.

 UART is a hardware component for translating the data between parallel and serial interfaces.

 UART does convert bytes of data to and from asynchronous start stop bit.

 UART is normally used in MODEM.

14. What does UART contain?

A clock generator, Input and Output start Registers, Buffers and Transmitter/Receiver control.

15. What is meant by HDLC?

 HDLC stands for “High Level Data Link Control”.

 HDLC is a bit oriented protocol.

 HDLC is a synchronous data Link layer.

16. Name the HDLC’s frame structure?

Flag	Address	Control	Data	FCS	Flag

17. List out the states of timer.

There are eleven states as follows

 Reset state
 Idle state
 Present state
 Over flow state
 Over run state
 Running state
 Reset enabled state / disabled
 Finished state
 Load enabled / disabled
 Auto reload enabled / disabled
 Service routine execution enabled / disabled

18. Name some control bit of timer.

 Timer Enable
 Timer start
 Up count Enable
 Timer Interrupt Enable

19. What is meant by status flag?

Status flag is the hardware signal to be set when the timer reaches zeros.

20. List out some applications of timer devices.

- Real Time clock
- Watchdog timer
- Input pulse counting
- TDM
- Scheduling of various tasks

21. State the special features on I2C.

- Low cost
- Easy implementation
- Moderate speed (upto 100 kbps).

22. What are disadvantages of I2C?

- Slave hardware does not provide much support
- Open collector drivers at the master leads to be confused

23. What are the two standards of USB?

- USB 1.1
- USB 2.0

24. Draw the data frame format of CAN.

Start	Arbitration field	Control field	Data field	CRC field	Acknowledgement field	End of frame
1	12	6	0-64	16	2	7

25. What is the need of Advanced Serial High Speed Buses?

If the speed in the rate of 'Gigabits per second' then there is a need of Advanced Serial High Speed Buses.

26. What is meant by ISA?

- ISA stands for Industry standard Architecture.
- Used for connecting devices following IO addresses and interrupts vectors as per IBM pc architecture.

27. What is meant by PCI-X?

- PCI X offers more speed over PCI.
- 30 times more speed than PCI.

28. Define CPCI.

- CPCI stands for Compact peripheral component Interfaces.
- CPCI is to be connected via a PCI.
- CPCI is used in the areas of Telecommunication Instrumentation and data communication applications.

29. Define half-duplex communication.

Transmission occurs in both the direction, but not simultaneously.

30. Define full duplex communication.

Transmission occurs in both the direction, simultaneously

31. Define Real Time Clock (RTC).

Real time clock is a clock which once the system stats does not stop and can't be reset and its count value can't be reloaded.

32. Define Time-out or Time Overflow.

A state in which the number of count inputs exceeded the last acquirable value and on reaching that state, an interrupt can be generated.

33. Why do we need at least one timer in an ES?

The embedded system needs at least on timer device. It is used as a system clock.

16 MARKS

1. Explain the parallel port devices.

Parallel Port I/O devices

In this communication any number of ports could be connected with the device and the data communication is bidirectional in nature.

1. Single Bit Input and Output

- Parallel Port Single Bit Input
- Parallel Port Single Bit Output

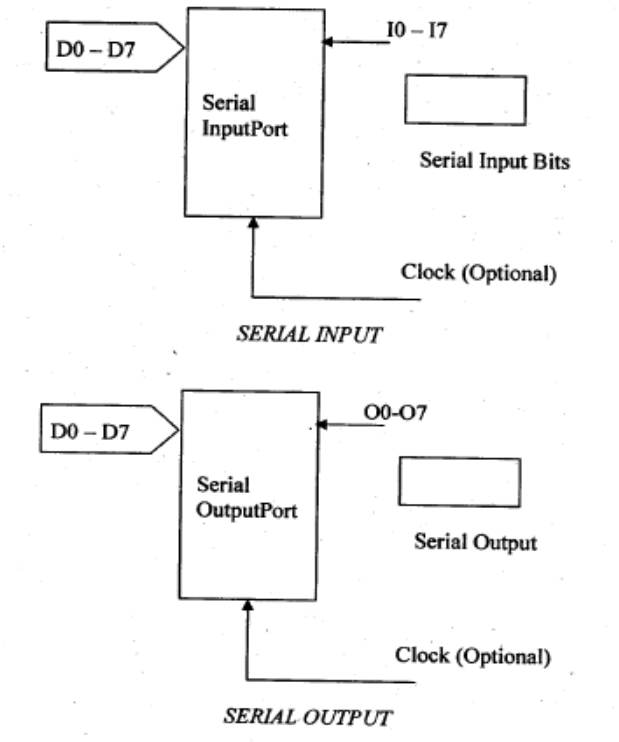
2. Parallel Port input and Output

- Parallel Port Input
- Parallel Port Output

Diagrams:

i) Parallel input port, output port and a bidirectional port for connecting the device

ii) The handshaking signals when used by the I/O ports.



Characteristics taken into consideration when interfacing a device port.

Synchronous Serial I/O devices

Synchronous Serial communication is defined as a Byte or a Frame of data is transmitted or received at constant time intervals with uniform phase differences.

 Synchronous Serial Input Devices
 Synchronous Serial Output Devices

2. Explain the sophisticated interfacing features in device ports.

 Schmitt trigger
 Data Gate
 HSTL,SSTL
 XCITE
 Multigigabyte Transceivers
 SerDes
 Multiple I/O standards
 PCS
 PMA

3. Explain the types of UART

 UART - Universal Asynchronous Receiver Transmitter
 USART Universal Synchronous Asynchronous Receiver Transmitter

Synchronous Serial Transmission

Synchronous serial transmission requires that the sender and receiver share a clock with one another, or that the sender provides a strobe or other timing signal so that the receiver knows when to "read the next bit of the data. A form of synchronous transmission is used with printers and fixed disk devices in that the data is sent on one set of wires while a clock or strobe is sent on a different wire. Printers and fixed disk devices are not normally serial devices because most fixed disk interface standards send an entire word of data for each clock or strobe signal by using a separate wire for each bit of the word.

Asynchronous Serial Transmission

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead the sender and receiver must agree on timing parameters in advance and special bits are added to each word which is used to synchronize the sending and receiving units.

 When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted.

 After the Start Bit, the individual bits of the word of data are sent, with the Least Significant Bit (LSB) being sent first.

 Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits and the receiver look at the wire at approximately through the assigned to each bit to determine if the bit is a 0 or 1.

 The sender does not know when the receiver has "looked" at the value of the bit. The sender only knows when the clock says to begin transmitting the next bit of the word.

 When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates.

 The Parity Bit may be used by the receiver to perform simple error checking. Then atleast one Stop Bit is sent by the transmitter.

When the receiver has received all of the bits in the data word. It may check for the Parity Bits (both sender and receiver must agree on whether a Parity Bit is to be used) and then the receiver looks for a Stop Bit. If the Stop Bit does not appear when it is supposed to the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted. Regardless of whether that data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host.

4. Describe in detail about Synchronous, ISO-Synchronous and Asynchronous communication for serial device.

Synchronous

In this means of communication, byte or frame of data received and transmitted at constant time intervals with uniform phase differences. Bits of a data frame are sent in a fixed maximum time intervals. Handshaking between sender and receiver is not provided during communication.

Example- Frames sent over LAN.

Characteristics

The main features of the synchronous communication are

 Bytes maintain a constant phase difference. No sending of bytes at random time intervals.

 A clock must be present at transmitter to send the data Moreover, the clock information is sent to the receiver (i.e.) it is not always implicit to the receiver.

Communication Protocols used

 Most often synchronous serial communication is used for data transmission between physical devices.

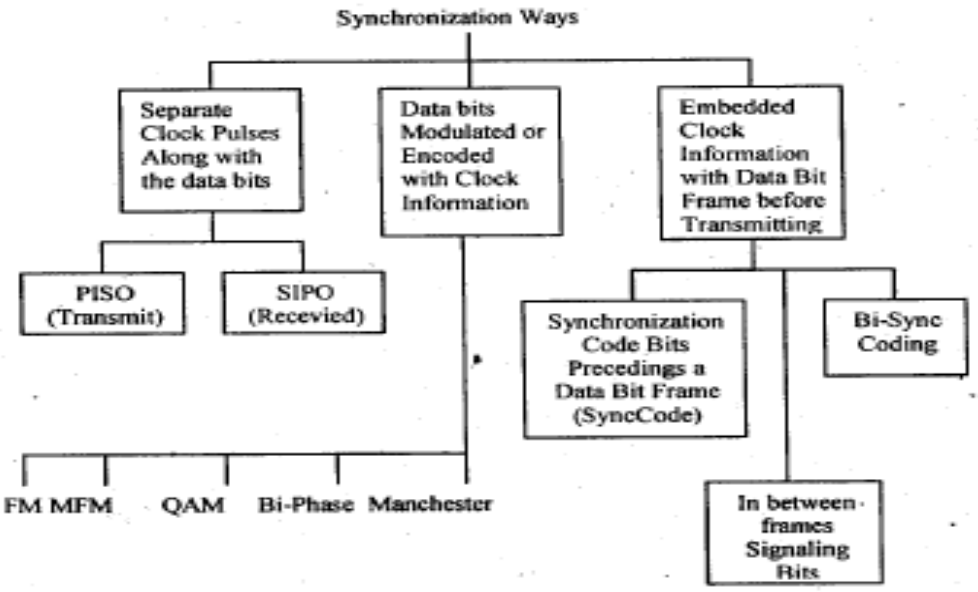
 It can be complex and has to be as per the communication

 Protocol followed.

 Example., HDLC (High Level Data Link Control)

Synchronization ways

Ten ways by which the synchronous signals with the clocking info transmitted from transmitter to the receiver are as shown below.



Iso-Synchronous

Iso-synchronous communication is a special case of synchronous communication. In contrast with the synchronous communication where bits of data frame are sent in a fixed maximum time interval, the iso=synchronous communication may have varied maximum time intervals.

Asynchronous

In the asynchronous communication a Byte or a Frame of data is received or sent at variable time intervals with phase difference. The data is sent as a series of bits. A shift register (in either hardware or software) is used to serialize each information byte into the series of bits which are then sent on the wire using an I/O port and a bus driver to connect to the cable

Characteristics

 Bytes or Frames of data is sent or received at variable time intervals.
 Handshaking between sender and receiver is provided during communication.
 A clock is needed at the transmitter to send the data'
 The clock data is not sent to the receiver (i.e) it is always ' implicit to the receiver.

5. Give some examples of Internal Serial Communication.

a. Common USART like Device in 8051

 There will be a common USART like hardware device in 805 1.
 USART - Universal Synchronous and Asynchronous.
 Receiver and Transmitter
 It is also called as SI (Serial Interface)

Features

i. SCON - Saves Control and status flags in SI
ii. SFR - Special Function Register
iii. SBUF - Serial Buffer

SI Operates in two modes

i. Half Duplex Synchronous mode of operation

ii. Full Duplex Asynchronous mode of operation

b. SPI and SCI: Serial Peripheral Interface (SPl)

 It has full duplex feature for asynchronous communication

 It has a feature of programmable rates for clock bits.

 It is also programmable for defining the –instance of the occurrence to negative and positive edges within intervals of bits. Devices selection is also programmable

Serial Communication Interfaces (SCI)

 UART asynchronous (SCD baud rates are same as SPI but not programmable.

 Communication is in full duplex

Characteristics

 A port device may have multi byte data input buffer and output buffer

 A port may have a DDR.

 Port may have LSTTL driving capability and port loading capability

 Multiple functionality in ports

Iso-synchronous communication is a special case of synchronous communication. In contrast with the synchronous communication where bits of data frame are sent in a fixed maximum time interval, the iso-synchronous communication may have varied maximum time intervals.

Protocol

Most often synchronous serial communication is used for

 Data transmission between physical devices.

 It can be complex and has to be as per the communication

 Protocol followed.

 Example., HDLC (High Level Data Link Control)

6. Explain Memory & IO Devices Interfacing (Memory Mapped I/O).

 I/O operations are interpreted differently depending on the viewpoint taken and place different requirements on the level of understanding of the hardware details.

 From the perspective of a system software developer, I/O operations imply communicating with the device, programming the device to initiate an I/O request, performing actual data transfer between the device and the system, and notifying the requestor when the operation completes. The system software engineer must understand the physical properties, such as the register definitions, and access methods of the device. Locating the correct instance of the device is part of the device communications when multiple instances of the same device are present.

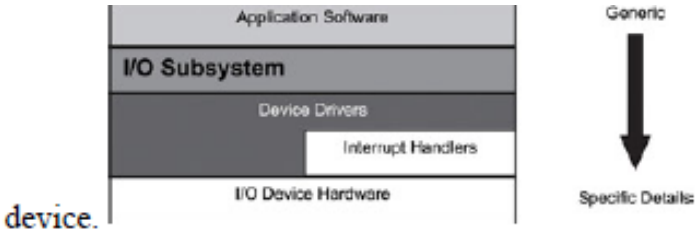
 The system engineer is also concerned with how the device is integrated with rest of the system. The system engineer is likely a device driver developer because the system engineer must know to handle any errors that can occur during the I/O operations.

 From the perspective of the RTOS, I/O operations imply locating the right device for the I/O request, locating the right device driver for the device, and issuing the request to the device driver. Sometimes the RTOS is required to ensure synchronized access to the device.

 The RTOS must facilitate an abstraction that hides both the device characteristics and specifics from the application developers. From the perspective of an application developer, the goal is to find a simple, uniform, and elegant way to communicate with all types of devices present in the system. The application developer is most concerned with presenting the data to the end user in a useful way.The combination of I/O devices, associated device drivers, and the I/O subsystem comprises the overall I/O system in an embedded environment.

 The purpose of the I/O subsystem is to hide the device-specific information from the kernel as well as from the application developer and to provide a uniform access method to the peripheral I/O devices of the system.

 This section discusses some fundamental concepts from the Perspective of the device driver developer. Figure 12.1 illustrates the I/O subsystem in relation to the rest of the system in a layered software model. As Shown, each descending layer adds additional detailed information to the architecture needed to manage a given device.



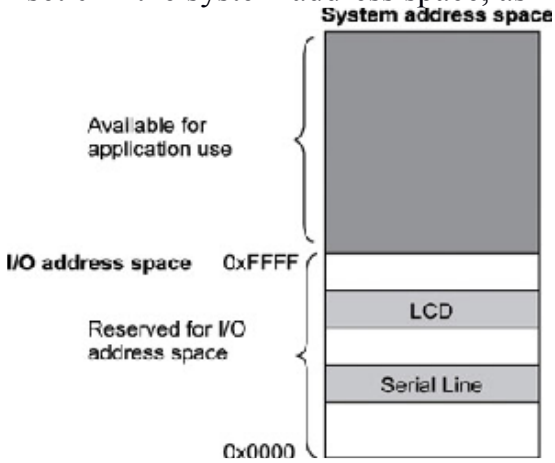
device.

Figure 12.1: I/O subsystem and the layered model.

 In memory-mapped I/O, the device address is part of the system memory address space. Any machine instruction that is encoded to transfer data between a memory location and the processor or between two memory locations can potentially be used to access the I/O device.

 The I/O device is treated as if it were another memory location. Because the I/O address space occupies a range in the system memory address space, this region of the memory address space is not available for an application to use.

 Figure 12.3: Memory-mapped I/O. The memory-mapped I/O space does not necessarily begin at offset 0 in the system address space, as illustrated in Figure



UNIT-III
PROCESS AND OPERATING SYSTEMS
2 MARKS

1. What are the states of a process?

a. Running
b. Ready
c. Waiting

2. What is the function in steady state?

Processes which are ready to run but are not currently using the processor are in the 'ready' state.

3. Define scheduling.

This is defined as a process of selection which says that a process has the right to use the processor at given time.

4. What is scheduling policy?

It says the way in which processes are chosen to get promotion from ready state to running state.

5. Define hyper period.

It refers the duration of time considered and also it is the least common multiple of all the processes.

6. What is schedulability?

It indicates any execution schedule is there for a collection of process in the system's functionality.

7. What are the types of scheduling?

1. Time division multiple access scheduling.
2. Round robin scheduling.

8. What is cyclostatic scheduling?

In this type of scheduling, interval is the length of hyper period 'H'. For this interval, a cyclostatic schedule is separated into equal sized time slots.

9. Define round robin scheduling.

This type of scheduling also employs the hyperperiod as an interval. The processes are run in the given order.

10. What is scheduling overhead?

It is defined as time of execution needed to select the next execution process.

11. What is meant by context switching?

The actual process of changing from one task to another is called a context switch.

12. Define priority scheduling.

A simple scheduler maintains a priority queue of processes that are in the runnable state.

13. What is rate monotonic scheduling?

Rate monotonic scheduling is an approach that is used to assign task priority for a preemptive system.

14. What is critical instant?

It is the situation in which the process or task posses' highest response time.

15. What is critical instant analysis?

It is used to know about the schedule of a system. Its says that based on the periods given,the priorities to the processes has to be assigned.

16. Define earliest deadline first scheduling.

This type of scheduling is another task priority policy that uses the nearest deadline as the criterion for assigning the task priority.

17. What is IDC mechanism?

It is necessary for a 'process to get communicate with other process' in order to attain a specific application in an operating system.

18. What are the two types of communication?

1. Blocking communication 2. Non-blocking communication

19. Give the different styles of inter process communication.

1. Shared memory.
2. Message passing.

16 MARKS

1. Explain Multiple Tasks and Multiple Processes.

 Many embedded computing systems do more than one thing that is, the environment can cause mode changes that in turn cause the embedded system to behave quit differently.

 The text compression box provides a simple example of rate control problems. A control panel on a machine provides an example of a different type of rate control problem, the asynchronous input.

 Multirate embedded computing systems are very common, including automobile engines, printers, and telephone PBXs.

 The co-routine was a programming technique commonly used in the early days of embedded computing to handle multiple processes.

 The ARM code in the co-routines is not intended to represent meaningful computations.

 The co-routine structure lets us implement more general kinds of flow of control than is possible with only subroutines; the identification of co-routine entry points provides us with some hooks for nonhierarchical calls and returns within the program.

 However, the co-routine does not do nearly enough to help us construct complex programs with significant timing properties.

 The co-routine in general does very little to simplify the design of code that satisfies timing requirements.

2. Explain Context Switching.

The context switch is the mechanism for moving the CPU from one executing process to another.

 Clearly, the context switch must be bug-free-a process that does not look at a real-time clock should not be able to tell that it was stopped and then restarted.

 Cooperative multitasking-the most general form of context switching, preemptive multitasking.
 Preemptive Multitasking-the interrupt is an ideal mechanism on which to build context switching for preemptive multitasking.
 A timer generates periodic interrupts to the CPU.
 The interrupt handler for the timer calls the operating system, which saves the previous process's state in an activation record, selects the next process to execute, and switches the context to that process.
 Processes and Object-Oriented Design-UML often refers to processes as active objects,that is, objects that have independent threads of control.
 We can implement the preemptive context switches using the same basic techniques.
 The only difference between the two is the triggering event, voluntary release of the CPU in the case of cooperative and timer interrupt in the case of preemptive.

3. Explain Scheduling policies.

 A scheduling policy defines how processes are selected for promotion from the ready state to the running state.
 Utilization is one of the key metrics in evaluating a scheduling policy.
 Rate-monotonic scheduling (RMS), introduced by Liu and Layland [Liu73], was one of the first scheduling policies developed for real-time systems and is still very widely used.
 The theory underlying RMS is known as rate-monotonic analysis(RMA).
 Earliest deadline first (EDF) is another well-known scheduling policy. It is a dynamic priority scheme-it changes process priorities during execution based on initiation times.
 RMS VERSUS EDF-EDF can extract higher utilization out of the CPU, but it may be difficult to diagnose the possibility of an imminent overload.
 A Closer Look at Our Modeling Assumptions-Our analyses of RMS and EDF have made some strong assumptions.
 Other POSIX Scheduling Policies-In addition of SCHED_FIFO,POSIX supports two other real-time scheduling policies: SCHED_RR and SCHED_OTHER.
 The SCHED_OTHER is defined to allow non-real-time processes to intermix with realtime processes.

4. Explain Inter process Communication Mechanism.

 Signals-Unix supports another, very simple communication mechanism-the signal.
 A signal is simple because it does not pass data beyond the existence of the signal itself.
 Signals in UML-A UML signal is actually a generalization of the Unix signal. While a Unix signal carries no parameters other than a condition code,a UML signal is an object.
 Shared Memory Communication-Conceptually, semaphores are the mechanism we use to make shared memory safe.

 POSIX supports semaphores, but it also supports a direct shared memory mechanism.
 POSIX supports counting semaphores in the _POSIX_SEMAPHORES option. A counting semaphore allows than one process access to a resource at a time.
 Message-Based Communication:-The shell syntax of the pipe is very familiar to Unix users. An example appears below.
 % foo file1| baz > file2
 A parent process use the pipe() function to create a pipe to talk to a child. It must do so before the child itself is created or it won't have any way to pass a pointer to the pipe to the child.
 The pipe () function returns an array of file descriptors, the first for the write end and the second for the read end.

5. Explain Shared Memory Communication and Message-Based Communication.

 Shared Memory Communication:-Conceptually, semaphores are the mechanism we use to make shared memory safe.
 POSIX supports semaphores, but it also supports a direct shared memory mechanism.
 POSIX supports counting semaphores in the _POSIX_SEMAPHORES option.
 A counting semaphore allows more than one process to a resourse at a time.
 If the semaphore allows up to N resources, then it will not block until N processes have simultaneously passed the semaphore.
 Message-Based Communication:-

o The shell syntax of the pipe is very familiar to Unix users. An example appears below % foo file1 | baz > file2

o POSIX also supports message queues under the _POSIX_MESSAGE_PASSING facility. The advantage of a queue over a pipe is that, since queues have names, we don't have to create the pipe descriptor before creating the other process using it, as with pipes.

UNIT IV
HARDWARE ACCELERATES & NETWORKS
2 MARKS

1. Name the important terms of RTOS.

Task
State
Scheduler
Shared data
Reentrancy

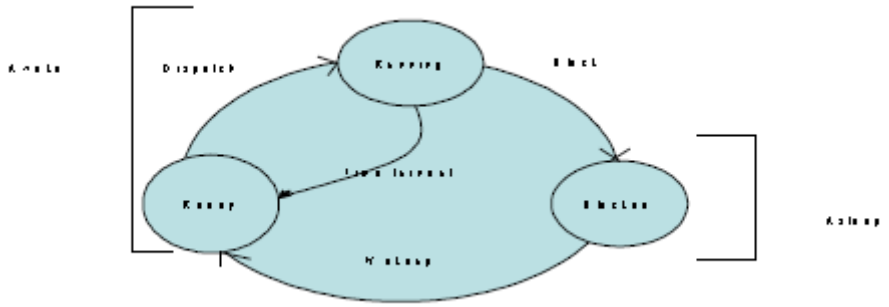
2. Define process.

Process is a computational unit that processes on a CPU under the control of a scheduling kernel of an OS. It has a process structure, called Process control block. A process defines a sequentially executing program and its state.

3. What is meant by PCB?

Process Control Block' is abbreviated as PCB.PCB is a data structure which contains all the information and components regarding with the process.

4. Draw the process state transitions.



5. Define task and Task state.

A task is a set of computations or actions that processes on a CPU under the control of a scheduling kernel. It also has a process control structure called a task control block that saves at the memory. It has a unique ID. It has states in the system as follows: idle, ready, running, blocked and finished

6. Define Task Control Block (TCB)

A memory block that holds information of program counter, memory map, the signal dispatch table, signal mask, task ID, CPU state and a kernel stack.

7. What is a thread?

Thread is a concept in Java and UNIX and it is a light weight sub process or process in an application program. It is controlled by the OS kernel. It has a process structure, called thread stack, at the memory. It has a unique ID .It have states in the system as follows: stating, running, blocked and finished.

8. Define Inter process communication.

An output from one task passed to another task through the scheduler and use of signals, exception, semaphore, queues, mailbox, pipes, sockets, and RPC.

9. What is shared data problem?

If a variable is used in two different processes and another task if interrupts before the operation on that data is completed then the value of the variable may differ from the one expected if the earlier operation had been completed .This ids known as shared data problem.

10. Define Semaphore.

Semaphore provides a mechanism to let a task wait till another finishes. It is a way of synchronizing concurrent processing operations. When a semaphore is taken by a task then that task has access to the necessary resources. When given the resources unlock. Semaphore can be used as an event flag or as a resource key.

11. Define Mutex.

A phenomenon for solving the shared data problem is known as semaphore. Mutex is a semaphore that gives at an instance two tasks mutually exclusive access to resources.

12. Differentiate counting semaphore and binary semaphore.

Binary semaphore

When the value of binary semaphore is one it is assumed that no task has taken it and that it has been released. When the value is 0 it is assumed that it has been taken.

Counting semaphore

Counting semaphore is a semaphore which can be taken and given number of times.Counting semaphores are unsigned integers.

13. What is Priority inversion?

A problem in which a low priority task inadvertently does not release the process for a higher priority task.

14. What is Deadlock situation?

A set of processes or threads is deadlocked when each process or thread is waiting for a resource to be freed which is controlled by another process.

15. Define Message Queue.

A task sending the multiple FIFO or priority messages into a queue for use by another task using queue messages as an input.

16. Define Mailbox and Pipe.

A message or message pointer from a task that is addressed to another task.

17. Define Socket.

It provides the logical link using a protocol between the tasks in a client server or peer to peer environment.

18. Define Remote Procedure Call.

A method used for connecting two remotely placed methods by using a protocol. Both systems work in the peer to peer communication mode and not in the client server mode.

19. What are the goals of RTOS?

 Facilitating easy sharing of resources
 Facilitating easy implantation of the application software
 Maximizing system performance
 Providing management functions for the processes, memory, and I/Os and for other functions for which it is designed.
 Providing management and organization functions for the devices and files and file like devices.
 Portability
 Interoperability
 Providing common set of interfaces.

20. What is RTOS?

An RTOS is an OS for response time controlled and event controlled processes. RTOS is an OS for embedded systems, as these have real time programming issues to solve.

21. List the functions of a kernel.

 Process management
 Process creation to deletion

 Processing resource requests
 Scheduling
 IPC
 Memory management
 I/O management
 Device management

22. What are the two methods by which a running requests resources?

 Message
 System call

23. What are the functions of device manager?

 Device detection and addition
 Device deletion
 Device allocation and registration
 Detaching and deregistration
 Device sharing

24. List the set of OS command functions for a device.

 Create and open
 Write
 Read
 Close and delete

25. List the set of command functions of POSIX file system

Open
Write
Read
Seek
Close

26. What are the three methods by which an RTOS responds to a hardware source call on interrupt?

 Direct call to ISR by an interrupt source
 Direct call to RTOS by an interrupt source and temporary suspension of a scheduled task.
 Direct call to RTOS by an interrupt source and scheduling of tasks as well as ISRs by the RTOS.

27. Name any two important RTOS.

 MUCOS
 VxWorks

28. Write short notes on Vxworks?

 Vxworks is a popular Real-time multi-tasking operating system for embedded microprocessors and systems.
 Vxworks can run on many target processors.
 It is a UNIX like Real time operating system.
 More Reliable
 More faster

29. What is meant by well tested and debugged RTOS?

An RTOS which is thoroughly tested and debugged in a number of situations.

30. What is sophisticated multitasking embedded system?

A system that has multitasking needs with multiple features and in which the tasks have deadlines that must be adhered to.

31. What are the features of UC/OS II?

Preemptive, Portable, Scalable, Multitasking

32. What is MICRO C/OS II?

- It stands for micro-controller operating system(UC/OS II).
- It is a real time kernel
- The other names of MICROC/OS II are MUCOS and UCOS.
- The codes are in 'C' and Assembly language.

33. What are the real time system level functions in UC/OS II? State some?

1 Initiating the OS before starting the use of the RTOS fuctions.
2 Starting the use of RTOS multi-tasking functions and running the states.
3 Starting the use of RTOS system clock.

34. Write the interrupt handling functions?

int connect ( ) is the function for handling the Interrupt.
int Lock ( ) -> Disable Interrupts.
int unlock( ) -> Enable functions.

35. Write down the seven task priorities in embedded 'C++'.

define Task _Read ports priority
define Task _Excess Refund priority
define Task _Deliver priority
define Task _Refund priority
define Task _Collect priority
define Task _Display priority
define Task _Time Date Display priority

36. Name any two mailbox related functions.

- OS_Event *OSMboxCreate(void *mboxMsg)
- Void *OSMboxAccept(OS_EVENT *mboxMsg)

37. Name any two queue related functions for the inter task communications.

- OS_Event OSQCreate(void **QTop,unsigned byte qSize)
- Unsigned byte OSQPostFront(OS_EVENT *QMsgPointer,void *qmsg)

38. How is Vx Works TCB helpful for tasks?

Provides control information for the OS that includes priority, stack size, state and options. CPU context of the task that includes PC, SP, CPU registers and task variables.

39. What are the various features of Vx Works?

- VxWorks is a scalable OS
- RTOS hierarchy includes timers, signals, TCP/IP sockets, queuing functions library, Berkeley ports and sockets, pipes, UNIX compatible loader, language interpreter, shell,debugging tools, linking loader for UNIX.

40. What is an active task in the context of Vx Works?

Active task means that it is in one of the three states, ready, running, or waiting.

41. What are the task service functions supported by Vx Works?

- taskSpawn()
- taskResume()
- taskSuspand()
- taskDelay()
- taskSuspand()
- taskInit()
- exit()
- taskDelete()

42. Name any four interrupt service functions supported by Vx Works.

- intLock()
- intVectSet()
- intVectGet()
- intContext()

43. Name some of the inter process communication function.

- semBCreate()
- semMCreate()
- semCCreate()
- semTake()
- semDelete()

44. Name some of the inter process communication function used for messaging.

msgQCreate(),msgQDelete(),msgQSend() and msgQReceive()

45. What are Vx Works pipes?

VxWorks pipes are queues that can be opened and closed like a pipe. Pipes are like virtual IO devices that store the messages as FIFO.

46. What are the different types of scheduling supported by Vx Works?

Preemptive priority and Time slicing

47. What are the task service functions supported by MUCOS?

- Void OSInit (void)
- Void OSStart(void)
- voidOSTickInit(void)
- void OSIntEnter(void)
- void OSIntExit(void)

48. What are the semaphores related functions supported by MUCOS?

- OS_Event OSSemCreate(unsigned short sem val)
- Void OSSemPend(OS_Event *eventPointer,unsigned short timeout,unsigned byte*SemErrPointer)
- unsigned short OSSemAccept(OS_Event*eventPointer)
- unsigned short OSSemPost(OS_Event*eventPointer)

16 MARKS

1. Explain the goals of operating system services.

Definition

An operating system (os) is Software that shares a computer system's resources (processor, memory, disk space, network, bandwidth and so on) between user's and tie application programs they run.

Goals

The OS goals are perfection and correctness to achieve the following

- To hide details of hardware by creating abstraction
- To allocate resources to process
- Effective user interface

Structure

When using an OS, the processor in the system runs in two modes.

User mode:

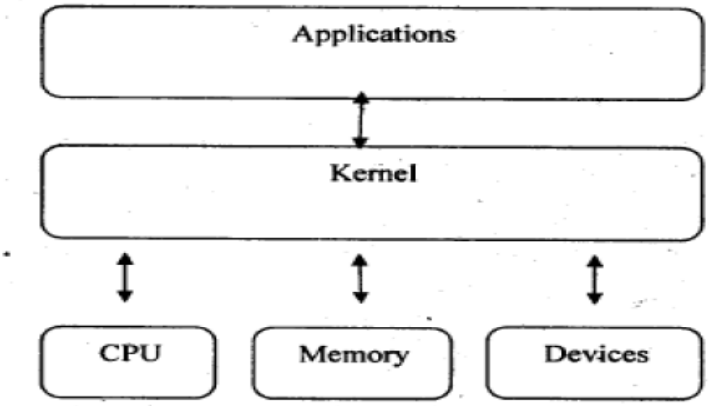
The user process is permitted to run and use only a subset of functions and instructions in the OS.

Supervisory mode:

The OS runs the privileged functions and instructions in protected mode and OS.A system can be assumed to have a startup as per below table

Kernel

The kernel is the central component of most computer operating systems (OS). Its responsibilities include managing the system's resources



Process management

Every program running on a computer, be it a-service or an application, is a process. Most operating systems enable concurrent execution of many processes and programs at once via multitasking, even with one CPU.

- On the most fundamental of computers multitask is done by simply switching process quickly.
- Depending on the operating system, as more processes run, either each time slice will become smaller or there will be a longer delay before each process is given a chance to run.
- Process management involves computing and distributing CPU time as well as other resources.

 Most operating systems allow a process to be assigned a priority which affects its allocation of CPU time.
 Interactive operating systems also employ some level of feedback in which the task with which the user is working receives higher priority.

Memory management

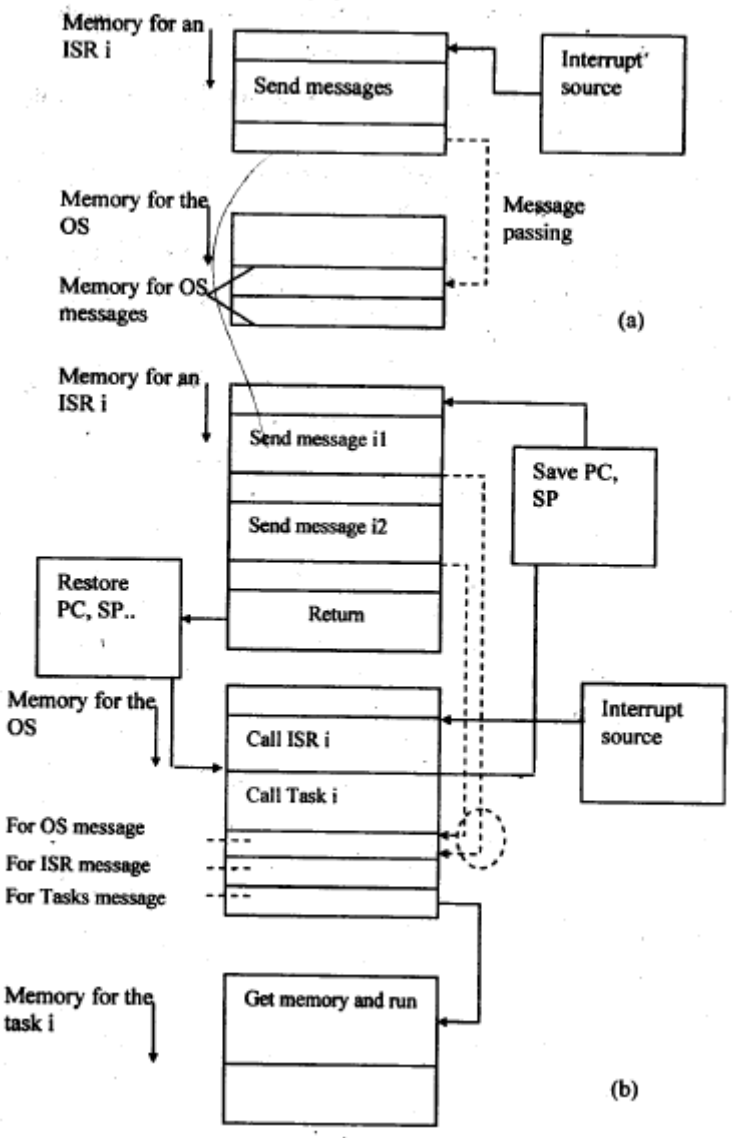
An operating system's memory manager coordinates the use of these various types of memory by tracking which can is available, which is to be allocated or deal located and how to move data between them. This activity, usually referred to as virtual memory management, increases the amount of memory available for each process by making the disk storage seem like main memory. There is a speed penalty associated with using disks or other slower' storage as memory - if running processes require significantly more RAM than is available, the system may start thrashing. This can happen either because one process requires a large amount or because two or more processes compete for a larger amount of memory than is available. This then leads to constant transfer of each process's data to slower storage. Another important part of memory management is managing virtual addresses.

2. Explain the three alternative systems in three RTOS for responding a hardware source call with the diagram.

There are three alternative systems for the RTOSs to respond to the hardware source calls from the interrupts.

Direct call to ISR by an Interrupting Source

 A hardware source calls an ISR directly, and ISR just sends a message to the RTOS. On an interrupt the process running at the CPU is interrupted and the ISR corresponding to that source starts executing.
 RTOS is simply sent a message from the ISR into a mailbox or message queue. It is to inform the RTOS about which ISR has taken control of the CPU.
 The ISR continues execution of the codes needed interrupt service. The routine sends a message for the RTOS.The message is stored at the memory allotted for RTOS messages.
 When TSR finishes, the RTOS returns to the interrupted process or reschedules the processes. RTOS action depends on the messages at the mailbox.

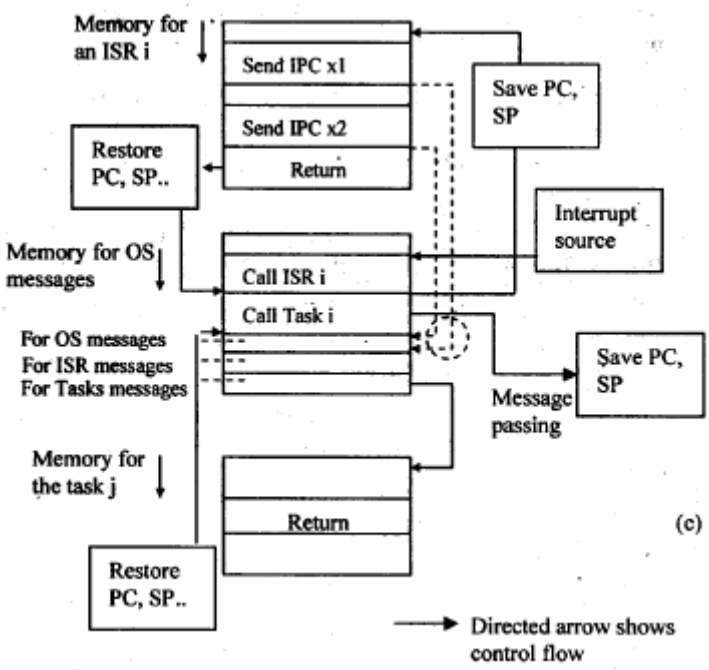


Direct call to RTOS by an Interrupting Source and temporary Suspension of a Scheduled Task

 The ISR send a message for initializing the task and returns after restoring the context. The messages are stored at the memory allotted for RTOS messages.

 The RTOS now initiates the task to ready state to later encode needed for the interrupt service. The ISR must be short and it simply places the messages. It is the task that runs the remaining codes whenever it is scheduled. RTOS schedules only-the processes) and switches the contexts between the tasks only.

 ISR executes only during a temporary suspension of a task.



Direct call to RTOS by an Interrupting Source and Scheduling of Tasks as well as IRS by the RTOS

 The RTOS intercepts and executes the task needed on return from the ISR without any message for the task from the ISR. This is shown below

 The routine doesn't T4 any messages but memory sends the IPCs for the needed parameters for a task. Parameters are stored at the memory allotted for the RTOS inputs.

 The RTOS now calls return and restoration of the context and switches the run later the codes needed for the any other task.

 The ISR need not be short and simply generates and saves as the IPCs the input parameters. It is the task that runs the codes whenever RTOS schedules not only the tasks but also the ISRs and switches the contexts as well as the tasks as well as the ISRs.

3. Explain the scheduler in which RTOS insert into the list and the ready task for sequential execution in a co-operative round robin model.

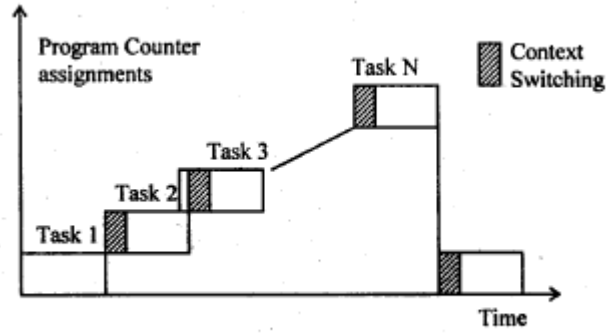
Cooperative means that each ready task cooperates to let a running one finish. None of the tasks does a block anywhere during the ready to finish states. Round robin means that each ready state runs in turn only from the circular queue. The service is in the order in which a task is initiated on interrupt.

 Worst-case latency is same for each task. It is tcycle

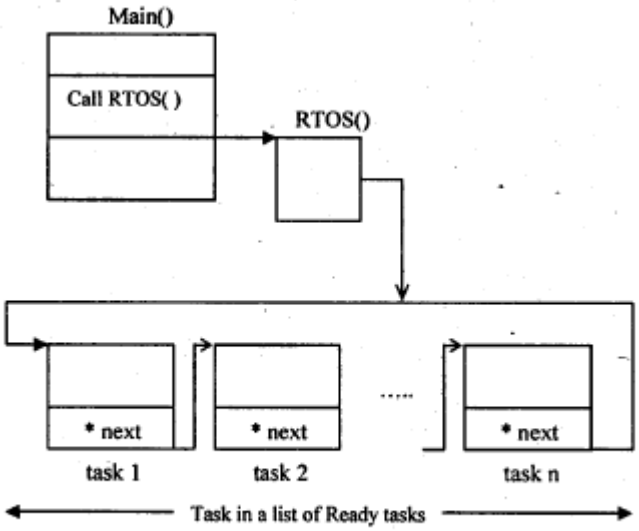
 The worst-case latency with this scheduling will be

$$T_{worst} = \{(dt_i + st_i + et_i)_1 + (dt_i + St_i + et_i)_2 + .. .+ (dt_i + st_i + et_i)_{n-1} + (dt_i + st_i + et_i)_n + t_{ISR}$$

dt- event detection time
st- switching time from one task to another
et -task execution time



(a)



(b)

4. Explain the use of Semaphores for a Task or for the Critical Sections of a Task.

Use of a Single Semaphore

Semaphore provides a mechanism to let a task wait till another finishes. It is a way of synchronizing concurrent process operations. When a semaphore is 'taken' by a task, then that task has to access to the necessary resources; when given, the resources unlock. Semaphore can be used as an event flag or as a resource key. Resource key is one that permits use of resources like CPU, memory or other functions or critical section codes. Semaphore, which is a binary Boolean variable (or it is a signaling variable or notifying variable.)Used as event flag. Semaphore is called binary semaphore when its value is 0 and it is assumed that it has Boolean taken. When its value is l, it is assumed that no task has taken it and that it has been released.

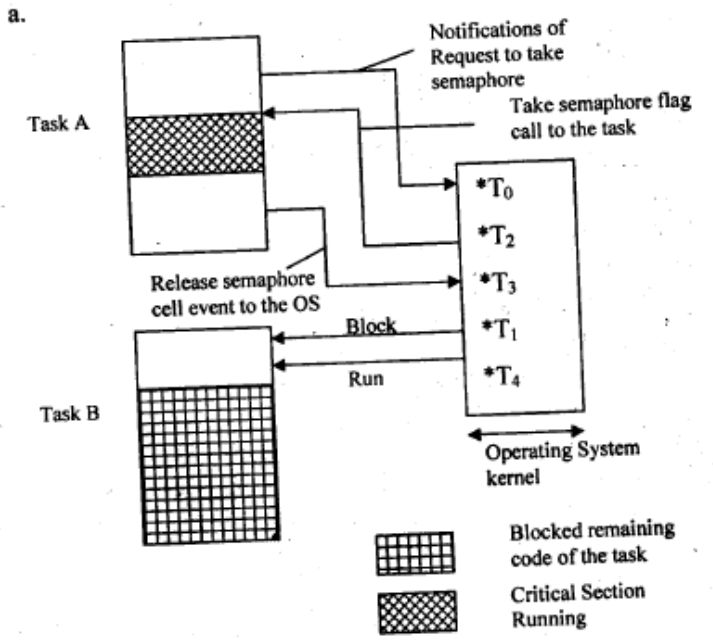


Figure 4.10 (a) shows the semaphore between tasks, A and B it shows the five sequential actions at five different instants

Use of Multiples Semaphores

Consider two semaphores. A task I when executing a critical section notifies the os to take the semaphore. os returns information that the semaphore has been taken to l and M. Now, the task I executes the codes of the critical. The OS having been notified about a take semaphore x from I, does not take and OS does not release the o task J and M' But the os returns an semaphore at an instance.

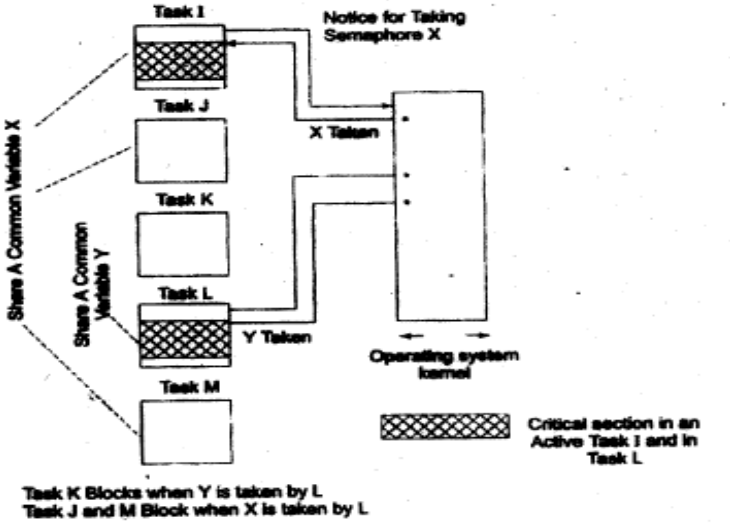


Figure 4.11 shows the use of two semaphores, x and y between the tasks, I to M.

Use of Mutex

Mutex is a semaphore that gives at an instance two tasks mutually exclusive access to resources. Use of mutex facilitates mutually exclusive access by two or more process to the resource (CPU). The same' variable, sem_m, is shared between the various processes. Let process and process 2 share sem_m and its initial value = l.

i) Process's 1 proceeds after sem_m decreases and equals 0 and gets the exclusive access to the CPU.

ii) Process I ends after sem_m increases and equals l; process 2 can now gets exclusive access to the CPU.

5. Explain the Rate Monotonic Co-operative scheduling.

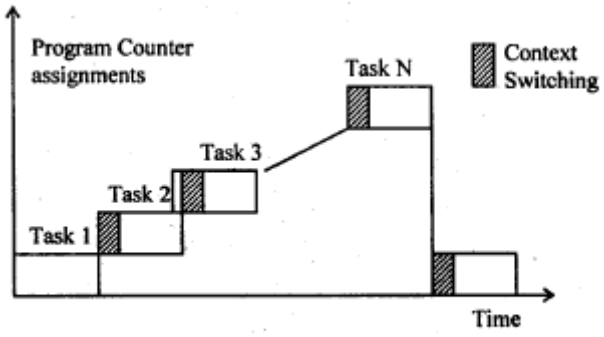
Cooperative means that each ready task cooperates to let a running one finish. None of the tasks does a block anywhere during the ready to finish states. Round robin means that each ready state runs in turn only from the circular queue. The service is in the order in which a task is initiated on interrupt.

- Worst-case latency is same for each task. It is tcycle .
- The worst-case latency with this scheduling will be:

$$T_{worst} = \{(dt_i + st_i + et_i)_1 + (dt_i + St_i + et_i)_2 + .. .+ (dt_i + st_i + et_i)_{n-1} + (dt_i + st_i + et_i)_n + t_{ISR}$$

t_{cycle}-a period for one round n the circular queue of ready tasks. The longer the queue, the greater is the tcycle.

dt- event detection time
st- switching time from one task to another
et -task execution time



(a)

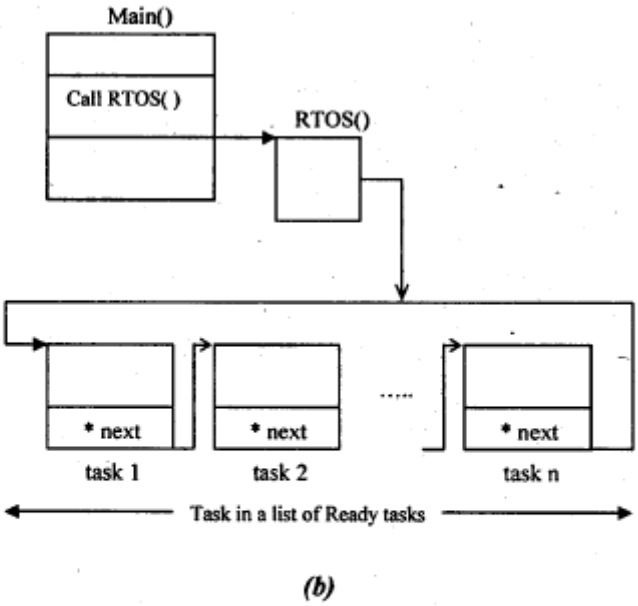


Figure 4.5) graph shows Program counter assignments (switch) at different times, when the scheduler calls the tasks one by one in the circular list and (b) shows a scheduler in which RTOS inserts into a list the ready tasks for a sequential execution in a cooperative round robin mode.

There is cooperative scheduling and each ready task cooperates to let the running one finish. None of the tasks does a block anywhere from the start to finish. Here, however, in case of cyclic scheduling, the round robin is among the ready tasks that run in turn only from a priority wise ordered list. The ordering is according to the precedence of interrupt source and the task.

 The RTOS scheduler first executes only the first task at the ordered list, and the tcycle equals the period taken by the first task on the list. It is deleted from the list after the first task is executed and the next task becomes the first. Now, if the task in the ordered list is executed in a cyclic order, it is called Cyclic Priority based cooperative Scheduler.

 The insertions and deletions for forming the ordered list are made only at the beginning of each cycle.

6. Explain the features of Vx Works.

VxWorks is a Unix-like real-time operating system made and sold by Wind River Systems of Alameda, California" USA. Like most RTOSes, VxWorks includes a multitasking kernel with pre-emptive scheduling and fast interrupt response, extensive inter-process communications and synchronization facilities, and a file system. Major distinguishing features of VxWorks include efficient POSlX-compliant memory management, microprocessor facilities, a shell for user interface, symbolic and source level debugging capabilities, and performance monitoring. VxWorks is generally used in embedded systems. Unlike "native" systems such as UNIX and Forth, VxWorks development is done on a "host"

machine running Unix or Windows, crosscompiling target software to run on various "target" CPU architectures as well as on the "host" by means of VxSim. VxWorks has been ported to a number of platforms and now runs on practically any modern CPU that is used in the embedded market. This includes the x 86 families, MIPS, PowerPC, SH-4 and the closely related family of ARM, StrongARM and xScale CPUs

VxWorks System Functions and System Tasks

The first task that a scheduler executes is UsrRoot from the entrv point of usrRoot0 in file install/Dir/target /config/alI / usr/Confi g. c.

It spawns the VxWorks tools and the following tasks. The root terminates after all the initializations. Any root task can be initialized or terminated. The set of functions, tlog Task, logs the system message without current task context I/O.

Interrupt handling functions

An internal hardware device auto generates an interrupt vector address, ISR-ECTADDR as per the device. Exceptions are defined in the us6r software.

ISR Design

 ISR have the highest priority and can preempt any running task.

 An ISR inhibits the execution of the tasks till return'.

 An IRS does not execute like a task and does not have regular task context. It has special interrupt context'

 While each task has its own stack" unless and otherwise not permitted -by a special architecture of a System or a processor

 An ISR should not wait for taking the semaphore or other IPC.

 ISR should just write the required data at the memory or post an IPC so that it has short codes and most of its functions, execute at tasks.

 ISR should not use flodting-point functions as these takes longer time to execute.

Signals and interrupt handling functions

Function 'void sigHandler(int sigNum)' declares a signal servicing routine for a signal identified by sigNum and a signal servicing routine registers a signal as follows:

Signal (sigNum, sigISR). The parameters that pass are the sigNum and signal servicing routine name, sigISR.A pointwer pSigCtx associates with the signal context. The signal xcontewxt savwes PC, SP, registers, etc. like an ISR context.

The signal ISR calls the following functions:

 Call taskRestart0, to restart the task which generated the sigNum.

 Call exit0 to terminate the task, which generated the sigNum.

 Call longiumpO. This results in starting the. Execution from a memory location.

7. Explain the RTOS programming tool MicroC/OS-II.

MicroC/O$II (commonly termed PC/O$II or mC/OSi-il), is a low-cost priories-based preemptive real time multitasking operating system kernel for microprocessors, written mainly in the C fodarnming language. It is mainly intended for use in embedded systems.

Ports

It has ports for most popular processors and boards in the market and is suitable for use in safety critical embedded systems such as aviation, medical systems and nuclear installations.

Task states

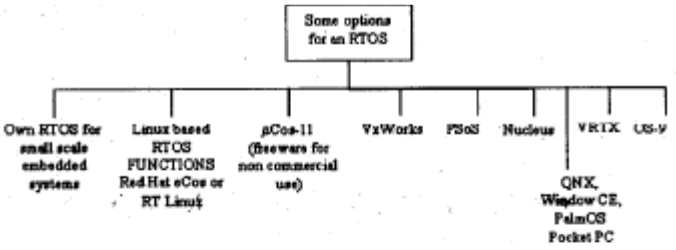
pC/OS-[is a multitasking operating system. Each task is an infinite loop and can be in any one of the following 5 states:

- Dormant
- Ready
- Running
- Waiting
- ISR

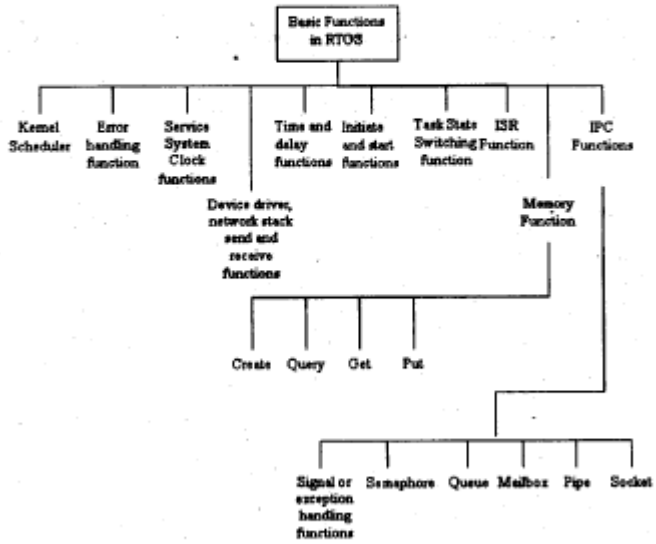
Source files

There are two types of source files.

Master header files includes the #include preprocessor commands for all the files of both types. It is referred to as include.h file Preprocessor dependent



a. Shows common options available for selecting an RTOS



b. Basic functions expected from kernel of an RTOS.

Figure 5.1 (a) shows common options available for selecting an RTOS, (b) Basic functions expected from kernel of an RTOS.

Source files

Two header files at the master are the following:

- os_cpu.h is the processor definition header file.
- The kernel building configuration file is os-cfg.h.
- Further two C file are the ISRs and RTOS timer
- specifying osjick.c and processor C codes os_cpu_c.c

Processor independent source file

Two files, MUCOS header and C files, are cos.ii.h and ucos.ii.c. The files for the RTOS core, timer and task are os_colt.c os_time.c and os_task.c. The other codes are in os_mem.c,os_sem.c, os_q.c and os_mbox.c

- It is a mandatory to use a well tested and debugged RTOS in a sophisticated multitasking embedded system.
- MUCOS and VxWorks are two important RTOS. MUCOS handles and schedules the task and ISRs.

8. Explain RTOS system level functions with an example.

MUCOS has system level functions. These are for RTOS initiation and start, RTC ticks initiation and the ISR enter and exit functions.

Prototype functions	When is this function called?
void OSInit(void)	At the beginning prior to OSStart()
void OSStart()	After the OSInit() and task creating functions
void OSTickInit(void)	Used in first task function that executes once to initialize the system timer ticks.
void OSIntEnter(void)	Just after the start of the ISR codes OSIntExit must call just before the return from the ISR.
void OSIntExit(void)	After the OSEnter() is called just after the start of the ISR codes and OSIntExit is called just before the return from the ISR.
OS_ENTER_CRITICAL	Macro to disable interrupts.
OS_EXIT_CRITICAL	Macro to enable interrupts.

Table 5.2 – RTOS initiate, start, and ISR functions for the tasks

Functions in this table pass no arguments and return type is void.

There is a global variable,OslntNesting.which increments on entering ISR.Global variable OSlntNesting decrements on'exit"from an ISR. Initiating the operating system before starting the use of the RTOS functions

- Function void OSInit (void) operating system.
- Its use is compulsory before functions.
- It returns no parameter.

Starting use of RTOS multitasking functions and returning the tasks

- Function void OSStart(void) is used to start the initiated operating system and

create tasks.
 Its use is compulsory for the multitasking OS kernel operations.
 It returns no parameter.

Starting the RTOS System clock

 Function void OsTicklnit(void) is used to initiate the system clock ticks and interrupts at regular intervals as per OS_TICKS_PER_SEC predefined during configuring the MUCOS.
 Its use is compulsory for the multitasking OS kernel operations when the timer functions ane to be used.
 It returns no parameter.

Sending message to RTOS taking control al the start of an ISR

 Function void OSlntEnter(void) is used at the start of an ISR.
 It is for sending a message to RTOS kernel for taking control. Its use is compulsory to let the multitasking OS kernel, control the nesting of the ISRs in case of occurrences of multiple interrupts of varying priorities.
 It returns no parameter.

UNIT V
CASE STUDY
2 MARKS

1. What is a PIC?

PIC refers to Programmable Intelligent Computer. PIC is microprocessor lies inside a personal computer but significantly simpler, smaller and cheaper. It can be used for operating relays, measuring sensors etc.

2. What are the main elements inside a PIC?

Processing engine, Program memory, data memory and Input/Output.

3. What are the types of program memory in a PIC?

Read-only, EPROM and EEPROM, Flash

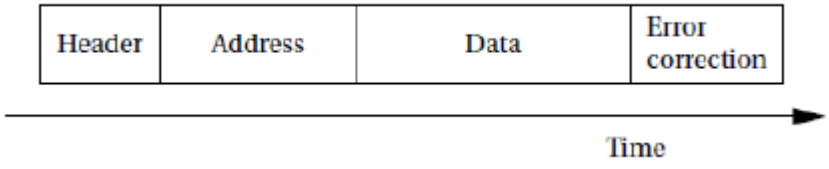
4. What is MBasic Compiler Software?

From version 5.3.0.0 onward, Basic Micro offers one version of its MBasic compiler, the “Professional” version. MBasic runs under Microsoft’s Windows operating system in any version from Windows 95 to Windows XP. The computer requires an RS-232 port for connection to the ISP-PRO programmer board.

5. Define pseudo-code.

Pseudo-code is a useful tool when developing an idea before writing a line of true code or when explaining how a particular procedure or function or even an entire program

6. Give the format of message on the bus.



7. What do you mean by software modem?

Low-cost modems generally use specialized chips, but some PCs implement the modem functions in software. The modem uses frequency-shift keying (FSK),a technique used in 1200-baud modems. Keying alludes to Morse code—style keying.

8. What do you mean by PDA?

A personal digital assistant (PDA), or handheld computer, is a small, mobile, handheld device that provides computing and information storage/retrieval capabilities.

9. What are the basic functions of PDA?

The vast majority of PDAs have five basic functions:

- o Contact management (names and addresses)
- o Scheduling (calendar)
- o Mobile phone functionality
- o To do list
- o Note-taking

10. What are the basic types of PDA?

• Clamshell PDAs have a small keyboard and they open out rather like a miniature laptop. They may also feature touch-sensitive screens and a stylus.

• Tablet-type PDAs do not have an integrated keyboard. Input occurs using a stylus or fold-away/portable keyboard and they tend to be palm-sized.

11. What do you mean by Set–Top–Box?

Set top boxes used to be just an analog cable tuner/decoder but now it includes the likes of digital cable, satellite controller, internet service controllers, digital video recording systems and home networking. These devices allow the various cable and satellite signal operators to deliver a wide variety of services from television to internet and the hardware manufacturers can provide many features and benefits including home networking capabilities.

12. Define System-On-Chip.

A system on a chip or system on chip (SoC or SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and often radio-frequency functions—all on a single chip substrate.

13. What is the difference between microcontroller and SOC?

Microcontrollers typically have under 100 kB of RAM (often just a few kilobytes) and often really are single-chip-systems, whereas the term SoC is typically used for more powerful processors, capable of running software such as the desktop versions of Windows and Linux, which need external memory chips (flash, RAM) to be useful, and which are used with various external peripherals.

14. What do you mean by FOSS?

Free and open-source software (FOSS) is computer software that can be classified as both free software and open source software. That is, anyone is freely licensed to use, copy, study, and change the software in any way, and the source

code is openly shared so that people are encouraged to voluntarily improve the design of the software.

15. What are the benefits of FOSS?

The benefits of using FOSS can include decreasing software costs, increasing security and stability (especially in regard to malware), protecting privacy, and giving users more control over their own hardware.

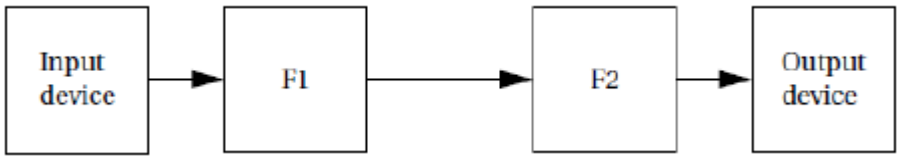
16 MARKS

1. Explain about hardware and software co-design.

Distributed embedded systems can be organized in many different ways depending upon the needs of the application and cost constraints. One good way to understand possible architectures is to consider the different types of interconnection networks that can be used.

 A point-to-point link establishes a connection between exactly two PEs. Point to point links is simple to design precisely because they deal with only two components. We do not have to worry about other PEs interfering with communication on the link.

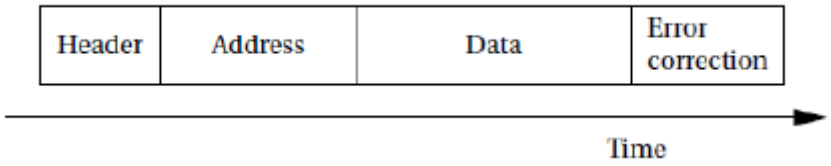
 Figure shows a simple example of a distributed embedded system built from point-to-point links. The input signal is sampled by the input device and passed to the first digital filter, F1, over a point-to-point link. The results of that filter are sent through a second point-to-point link to filter F2. The results in turn are sent to the output device over a third point-to-point link.



 A digital filtering system requires that its outputs arrive at strict intervals, which means that the filters must process their inputs in a timely fashion. Using point-to-point connections allows both F1 and F2 to receive a new sample and send a new output at the same time without worrying about collisions on the communications network.

 It is possible to build a full-duplex, point-to-point connection that can be used for simultaneous communication in both directions between the two PEs. (A half duplex connection allows for only one-way communication.)

 A bus is a more general form of network since it allows multiple devices to be connected to it. Like a microprocessor bus, PEs connected to the bus have addresses. Communications on the bus generally take the form of packets as illustrated in Figure. A packet contains an address for the destination and the data to be delivered.



 It frequently includes error detection/correction information such as parity. It also may include bits that serve to signal to other PEs that the bus is in use, such as the header shown in the figure.

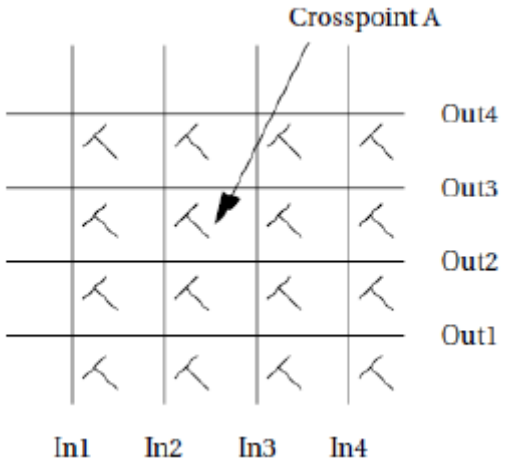
 The data to be transmitted from one PE to another may not fit exactly into the size of the data payload on the packet. It is the responsibility of the transmitting PE to divide its data into packets; the receiving PE must of course reassemble the complete data message from the packets.

 Distributed system buses must be arbitrated to control simultaneous access, just as with microprocessor buses. Arbitration scheme types are summarized below.

o Fixed-priority arbitration always gives priority to competing devices in the same way. If a high-priority and a low-priority device both have long data transmissions ready at the same time, it is quite possible that the low-priority device will not be able to transmit anything until the high-priority device has sent all its data packets.

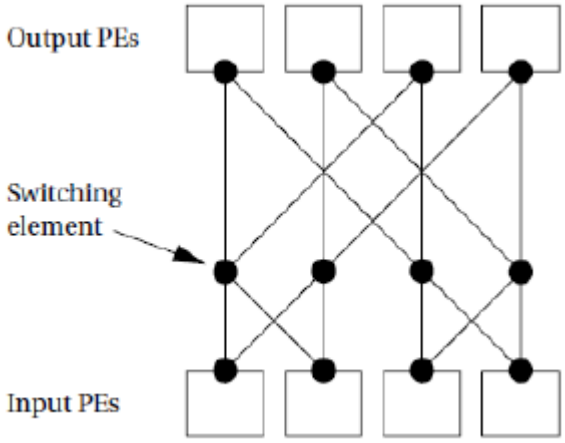
o Fair arbitration schemes make sure that no device is starved. Round-robin arbitration is the most commonly used of the fair arbitration schemes. The PCI bus requires that the arbitration scheme used on the bus must be fair, although it does not specify a particular arbitration scheme. Most implementations of PCI use round-robin arbitration.

 A bus has limited available bandwidth. Since all devices connect to the bus, communications can interfere with each other. Other network topologies can be used to reduce communication conflicts. At the opposite end of the generality spectrum from the bus is the crossbar network shown in Figure.



 A crossbar not only allows any input to be connected to any output, it also allows all combinations of input/output connections to be made. Thus, for example, we can simultaneously connect in1 to out4, in2 to out3, in3 to out2, and in4 to out1 or any other combinations of inputs.

 Many other networks have been designed that provide varying amounts of parallel communication at varying hardware costs. Figure shows an example multistage network. The crossbar is a direct network in which messages go from source to destination without going through any memory element.



 Multistage networks have intermediate routing nodes to guide the data packets.
 Most networks are blocking, meaning that there are some combinations of sources and destinations for which messages cannot be delivered simultaneously.
 A bus is a maximally blocking network since any message on the bus blocks messages from any other node. A crossbar is non-blocking.
 In general, networks differ from microprocessor buses in how they implement communication protocols. Both need handshaking to ensure that PEs do not interfere with each other. But in most networks, most of the protocol is performed in software.
 Microprocessors rely on bus hardware for fast transfers of instructions and data to and from the CPU. Most embedded network ports on microprocessors implement the basic communication functions (such as driving the communications medium) in hardware and implement many other operations in software.
 An alternative to a non-bus network is to use multiple networks. As with PEs, it may be cheaper to use two slow, inexpensive networks than a single high performance, expensive network.
 If we can segregate critical and noncritical communications onto separate networks, it may be possible to use simpler topologies such as buses. Many systems use serial links for low-speed communication and CPU buses for higher speed and volume data transfers.

2. Explain about data compressor.

 We use the Huffman coding technique, We require some understanding of how our compression code fits into a larger system.
 The data compressor takes in a sequence of input symbols and then produces a stream of output symbols. Assume for simplicity that the input symbols are one byte in length. The output symbols are variable length, so we have to choose a format in which to deliver the output data.

 Delivering each coded symbol separately is tedious, since we would have to supply the length of each symbol and use external code to pack them into words.
 On the other hand, bit-by-bit delivery is almost certainly too slow. Therefore, we will rely on the data compressor to pack the coded symbols into an array. There is not a one-to-one relationship between the input and output symbols, and we may have to wait for several input symbols before a packed output word comes out.
 The data compressor as discussed above is not a complete system, but we can create at least a partial requirements list for the module as seen below. We used the abbreviation N/A for not applicable to describe some items that do not make sense for a code module.
 Let's refine the description to come up with a more complete specification for our data compression module. That collaboration diagram concentrates on the steady-state behavior of the system.
 For a fully functional system, we have to provide the following additional behavior.
o We have to be able to provide the compressor with a new symbol table.
o We should be able to flush the symbol buffer to cause the system to release all pending symbols that have been partially packed. We may want to do this when we change the symbol table or in the middle of an encoding session to keep a transmitter busy.
 A class description for this refined understanding of the requirements on the module. The class's buffer and current-bit behaviors keep track of the state of the encoding, and the table attribute provides the current symbol table.
 The class has three methods as follows:
o Encode performs the basic encoding function. It takes in a 1-byte input symbol and returns two values: a boolean showing whether it is returning a full buffer and, if the boolean is true, the full buffer itself.
o New-symbol-table installs a new symbol table into the object and throws away the current contents of the internal buffer.
o Flush returns the current state of the buffer, including the number of valid bits in the buffer.
 The longest Huffman code for an eight-bit input symbol is 256 bits. (Ending up with a symbol this long happens only when the symbol probabilities have the proper values.)
 The insert function packs a new symbol into the upper bits of the buffer; it also puts the remaining bits in a new buffer if the current buffer is overflowed.
 The Symbol-table class indexes the encoded version of each symbol. The class defines an access behavior for the table; it also defines a load behavior to create a new symbol table.

3. Explain about software modem.

 Low-cost modems generally use specialized chips, but some PCs implement the modem functions in software. Before jumping into the modem design itself, we discuss principles of how to transmit digital data over a telephone line.

 The modem will use frequency-shift keying (FSK),a technique used in 1200-baud modems. Keying alludes to Morse code—style keying. The FSK scheme transmits sinusoidal tones, with 0 and 1 assigned to different frequencies.
 Sinusoidal tones are much better suited to transmission over analog phone lines than are the traditional high and low voltages of digital circuits. The 01 bit patterns create the chirping sound characteristic of modems. (Higher-speed modems are backward compatible with the 1200-baud FSK scheme and begin a transmission with a protocol to determine which speed and protocol should be used.)
 The scheme used to translate the audio input into a bit.The analog input is sampled and the resulting stream is sent to two digital filters (such as an FIR filter).
 One filter passes frequencies in the range that represents a 0 and rejects the 1-band frequencies, and the other filter does the converse.
 The outputs of the filters are sent to detectors, which compute the average value of the signal over the past n samples. When the energy goes above a threshold value, the appropriate bit is detected.
 We will send data in units of 8-bit bytes. The transmitting and receiving modems agree in advance on the length of time during which a bit will be transmitted (otherwise known as the baud rate). But the transmitter and receiver are physically separated and therefore are not synchronized in any way.
 The receiving modem does not know when the transmitter has started to send a byte. Furthermore, even when the receiver does detect a transmission, the clock rates of the transmitter and receiver may vary somewhat, causing them to fall out of sync. In both cases, we can reduce the chances for error by sending the waveforms for a longer time.
 The receiver will detect the start of a byte by looking for a start bit, which is always 0.
 By measuring the length of the start bit, the receiver knows where to look for the start of the first bit. However, since the receiver may have slightly misjudged the start of the bit, it does not immediately try to detect the bit.
 The modem will not implement a hardware interface to a telephone line or software for dialing a phone number. We will assume that we have analog audio inputs and outputs for sending and receiving. We will also run at a much slower bit rate than 1200 baud to simplify the implementation.
 Next, we will not implement a serial interface to a host, but rather put the transmitter’s message in memory and save the receiver’s result in memory as well. Given those understandings, let’s fill out the requirements table.
 The modem consists of one small subsystem (the interrupt handlers for the samples) and two major subsystems (transmitter and receiver).
 Two sample interrupt handlers are required, one for input and another for output, but they are very simple. The transmitter is simpler, so let’s consider its software architecture first.

 The best way to generate waveforms that retain the proper shape over long intervals is table lookup. Software oscillators can be used to generate periodic signals, but numerical problems limit their accuracy.
 An analog waveform with sample points and the C code for these samples. Table lookup can be combined with interpolation to generate high-resolution waveforms without excessive memory costs, which is more accurate than oscillators because no feedback is involved.
 The required number of samples for the modem can be found by experimentation with the analog/digital converter and the sampling code.
 The structure of the receiver is considerably more complex. The filters and detectors can be implemented with circular buffers. But that module must feed a state machine that recognizes the bits.
 The recognizer state machine must use a timer to determine when to start and stop computing the filter output average based on the starting point of the bit. It must then determine the nature of the bit at the proper interval. It must also detect the start bit and measure it using the counter.
 The receiver sample interrupt handler is a natural candidate to double as the receiver timer since the receiver's time points are relative to samples.
 The hardware architecture is relatively simple. In addition to the analog/digital and digital/analog converters, a timer is required. The amount of memory required to implement the algorithms is relatively small.

4. Explain about PDA and Set-Top-Box.

PERSONAL DIGITAL ASSISTANT:

 A personal digital assistant (PDA), or handheld computer, is a small, mobile, handheld device that provides computing and information storage/retrieval capabilities.
 The vast majority of PDAs have five basic functions:
o Contact management (names and addresses)
o Scheduling (calendar)
o Mobile phone functionality
o To do list
o Note-taking
 Many PDA manufacturers now include additional functionality in their products, such as:
o Access to the Internet
o The ability to play MP3 files
o The ability to read electronic books
o The ability to play games
o Bluetooth connectivity

Size and Weight

 The size and weight of PDAs can vary enormously – some are the size of a credit card, some fit in the palm of the hand and some are like UMPCs. Basic PDAs are confined to basic information organization, while the latest PDAs have

many additional functions and capabilities such as Bluetooth, Wi-Fi, GPS and extra memory storage options.

 Additional options one should consider purchasing include an external Bluetooth keyboard, an extra battery, docking cradle, travel synchronisation cable, extra stylus, case, USB/VGA cable for use with a data projector.

Broadly speaking, there are two types of PDA:

• Clamshell PDAs have a small keyboard and they open out rather like a miniature laptop. They may also feature touch-sensitive screens and a stylus.

• Tablet-type PDAs do not have an integrated keyboard. Input occurs using a stylus or fold-away/portable keyboard and they tend to be palm-sized.

SET-TOP-BOX:

 The continuing trend is to link broadband signal delivery to the home entertainment display, and other devices via set top boxes. Set top boxes used to be just an analog cable tuner/decoder but now it includes the likes of digital cable, satellite controller, internet service controllers, digital video recording systems and home networking.

 These devices allow the various cable and satellite signal operators to deliver a wide variety of services from television to internet and the hardware manufacturers can provide many features and benefits including home networking capabilities.

 There is digital video recording onto hard disk drives, replacing the cassette format, allowing pause and replay of and live television, or interactive TV. There are new standards being created to facilitate the design of the boxes such as a recent reference blueprint development by communications chipmaker Broadcom using the Microsoft interactive TV software system.

 The set top box is going to be a high volume commodity with many forms and functions.

5. Explain about system-on-silicon and FOSS tools.

SYSTEM-ON-SILICON:

A system on a chip or system on chip (SoC or SOC) is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and often radio-frequency functions—all on a single chip substrate. SoCs are very common in the mobile electronics market because of their low power consumption. A typical application is in the area of embedded systems.

The contrast with a microcontroller is one of degree. Microcontrollers typically have under 100 kB of RAM (often just a few kilobytes) and often really are single-chip-systems, whereas the term SoC is typically used for more powerful processors, capable of running software such as the desktop versions of Windows and Linux, which need external memory chips (flash, RAM) to be useful, and which are used with various external peripherals.

For larger systems, the term system on a chip is hyperbole, indicating technical direction more than reality: a high degree of chip integration, leading toward reduced manufacturing costs, and the production of smaller systems. Many

interesting systems are too complex to fit on just one chip built with a process optimized for just one of the system's tasks.

When it is not feasible to construct a SoC for a particular application, an alternative is a system in package (SiP) comprising a number of chips in a single package. In large volumes, SoC is believed to be more cost-effective than SiP since it increases the yield of the fabrication and because its packaging is simpler.

Another option, as seen for example in higher end cell phones and on the BeagleBoard, is package on package stacking during board assembly.

The SoC chip includes processors and numerous digital peripherals, and comes in a ball grid package with lower and upper connections.

The lower balls connect to the board and various peripherals, with the upper balls in a ring holding the memory buses used to access NAND flash and DDR2 RAM. Memory packages could come from multiple vendors.

FOSS TOOLS FOR EMBEDDED SYSTEM DEVELOPMENT:

Free and open-source software (FOSS) is computer software that can be classified as both free software and open source software. That is, anyone is freely licensed to use, copy, study, and change the software in any way, and the source code is openly shared so that people are encouraged to voluntarily improve the design of the software. This is in contrast to proprietary software, where the software is under restrictive copyright and the source code is usually hidden from the users.

The benefits of using FOSS can include decreasing software costs, increasing security and stability (especially in regard to malware), protecting privacy, and giving users more control over their own hardware.

Free, open-source operating systems such as Linux and descendents of BSD are widely utilized today, powering millions of servers, desktops, smart phones (e.g. Android), and other devices. Free software licenses and open-source licenses are used by many software packages.