ROEVER ENGINEERING COLLEGE

PERAMBALUR

DEPARTMENT OF ECE

SUBJECT NAME :EMBEDDED SYSTEMS

UNIT I

INTRODUCTION TO EMBEDDED SYSTEMS

Definition and classification – Overview of processors and hardware units in an embedded system – Software embedded into the system – Exemplary embedded systems – Embedded Systems on a Chip (SOC) and the use of VLSI designed circuits.

ANSWER KEYS
PART A

1. **Define: Embedded Systems**

   Computer hardware integrated with software for dedicated application.

2. **Give the classification of an embedded system.**

   Small scale- 8 or 16 bit microcontroller with little h/w and s/w,

   Medium scale - 16 or 32 bit micro controller, DSP or RISC processor

   Sophisticated – enormous h/w & s/w and programmable logic arrays

3. **What are the components of embedded system hardware?**

   Processor, timer, interrupt controller, display devices, I/O ports, power supply and required application circuits.

4. **What are the various forms of memories in the systems?**

   RAM, ROM, PROM, EEPROM, Flash and its explanation.

5. **What is GPIB?**

   General purpose interface bus follows IEEE 488 bus standard, used to link various instruments to the systems.

6. **Name important embedded processor chip used in industries?**

   Family name

   - ARM family ARM 7* and ARM 9*
   - INTEL family i960
   - AMD family 29050.

7. **Specify any four applications of embedded systems.**

   Smart card, telecom, missiles and satellites, automotive etc

8. **Explain the terms FPGA, CPLD, and RTOS and UART**

   FPGA- Field Programmable Gate array

   CPLD- Complex programmable Logic Device

   RTOS- Real Time Operating System

   UART-Universal Asynchronous Receiver and Transmitter

9. **What is watchdog timer?**

   Reset circuit, resets the system after the predefined time.

10. **What are the different models employed for embedded software design?**

    Finite state machine, Petrinet model, control and data flow graph, activity diagrams based UML model.

11. **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

12. **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.

13. **Give examples for general purpose processor.**

    - Microcontroller
    - Microprocessor

14. **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.

15. 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.

16. 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.

17. Define device driver.

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

18. 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.
-

19. 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
-

20. Give some examples for small scale embedded systems.

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

PART B

1. **(i) What are the Core processors employed in embedded systems and Explain in detail about microprocessor used in embedded systems. (12)**

   (ii) What are the important considerations in selecting a processor for embedded system design? (4)

. (i) Description about

- General purpose processor
- Application Specific system Processor
- Multiprocessor using GPP
- ASIC, VLSI, FPGA (2)
- Block diagram of basic Microprocessor (4)

Its operation, name of the other microprocessor and its features.

Units: ALU, register, buses, timing and control (6)

(ii) Instruction Set, Max. no. of bits, clock frequency, processor ability to solve complex algorithms (4)

2. **(i) Explain in detail about the power supply management in embedded system design (8)**

   (ii) Explain about the use of various reset circuits in processes (8)

(i) Power supply management (8)

Explanation of the following

- Ranges
- Precautions for connecting power supply rail
- Explanation for charge pump
- Low power consumption device
- Methods of avoiding power dissipation

(ii) Reset circuits (4)

Defn: To start executing instruction from the starting address

Reset Activating method:

- External reset circuits
- Software instructions

Watch dog timer: (4)

Defn: Timing device that resets the system after a predefined timeout

Explanation with an example

3. (i) Explain about the I/O ports & buses and its interfacing embedded systems (12)

(ii) What is the use of interrupt handler? (4)

. (i) I/O ports: (6)

Input port:

- Defn: The system gets inputs by the read operation at the port address
- Explanation with an example for that.

Output port:

- Defn: The system has output port through which it sends output bytes to the real world
- Explanation with suitable example.
- Types of I/O ports
    1. Serial – sends or receive the data in serial ie. Bit by bit sequence
    2. Parallel—sends or receive the data bits at parallel lines
    3. Explanation for both.

MUX and DEMUX:

- MUX – select only one output from 'n' number of inputs through the selection line
- DEMUX- the input is transferred to any one of the output lines through the selection line
- Explanation

I/O buses and interfaces (6)

- Types of buses
- I 2C, CAN, USB, ISA, EISA, PCI
- Features of the above Buses

(ii) Interrupts: (4)

- Defn Interrupt and its execution procedure

4. (i) What are the parameters to be considered in ADC application (4)

(ii) Explain the process of converting an ALP into machine implementable file **(12)**

(i) List and Explain the following Parameters of ADC (4)

- Max. Reference voltage
- Max. no. of bits
- Start conversion pulse
- End of conversion
- Sample & hold circuit.

(ii) Block diagram (5)

Explanation for the following (7)

- Assembler - it converts assembly software to machine code.
- Linker- it links the codes with the other required assembled code.
- Loader- it reallocate the codes after finding the physical RAM addresses
- Locater- the codes as ROM image are permanently placed in the available ROM
- Device programmer- it burns the ROM image with the inputs into the PROM or EPROM.

5. (i) Discuss about the contents of ROM image **(8)**

(ii) What are the different types of display devices in embedded system? **(8)**

(i) Memory allocation diagram (5)

Contents: (3)

Boot up program, stack(s), address pointer(s), program counter, address pointer, application tasks, ISRs, RTOS, input data and vector address

(ii) Explanation about LCD, LED array displays (8)

6. List the various hardware required in various exemplary embedded system with typical values. **(16)**

(i) Automatic chocolate vending machine (4)

Hardware required:

Processor- Micro controller, internal bus- 8 bit, processor architecture CISC , PROM- 4KB, RAM- 256 bytes on chip, I/O ports-input for coin sorter port, delivery port, display port.

(ii) ROBOT (4)

Hardware required:

Processor- Micro controller, internal bus- 8 bit, processor architecture CISC , PROM- 8KB, RAM- 256 bytes on chip, I/O ports-multiple ports for motors and for angle encoders, PWM for DAC,ADC

(iii) Mobile phone (4)

Hardware required:

Processor- Multi processor system on chip, internal bus- 32 bit, processor architecture RISC, caches and MMU, PROM- 1MB,EEPROM-32kb, RAM- 1MB on SoC, I/O ports-keypad and display ports, transceiver, real time detection of an event or signal, PWM for DAC, ADC, modulation demodulation, DSP instruction.

(iv)Voice processor (4)

Hardware required:

Processor- Microprocessor +DSP, internal bus- 32 bit, processor architecture RISC, caches and MMU, PROM- 1MB,EEPROM & Flash-4MB, RAM- 1MB off-chip, I/O ports-input port for speech and output port for replay, real time detection of an event or signal, PWM for DAC, ADC, DSP instruction.

- Or any four exemplary systems.

7. (i) What is SOC and explain it with an examples (8)

(ii) What are components of SOC smart card? (8)

(i) Defn- Designed on a single silicon chip (2)

Block diagram (3)

Explanation of the following: (3)

Two internal ASICs, ASIPs, shared memory, peripheral interfaces on a common bus.

(ii) Block diagram (4)

RAM, Rom, EEPROM, processor, timer & interrupt controller, ASK modulator, Charge pump Circuit, interfacing I/O, Transceiver antenna on silicon, power supply.

- Explanation of the above (4)

8. (i) List the various software tools of embedded systems and its uses (12)

(ii) Explain Software tools application with exemplary systems (4)

(i) Explanation for the following software tools: (10)

Editor , Interpreter, Compiler, Assembler, Cross assembler, Simulator, Source code, RTOS, Stethoscope, Trace scope, Integrated development , Environment Prototyper, Locater

(ii) Any two exemplary systems: (4)

Example (i): Robot

Editor, Interpreter, Assembler, Simulator, RTOS, Trace scope, Locater, Integrated development.

Example (ii): Mobile phone

Compiler, Simulator, Source code, RTOS, Stethoscope, Trace scope, integrated development, Environment Prototyper, Locater.

9. Draw and explain the functional block diagram of microcontroller based embedded system (16)

Block diagram (8)

Explanation of the following units. (8)

Processor, memories, I/O ports, data and stack in internal RAM, external memories, serial UART communication port, interrupt controller, ADC, DAC, DMA controller, Modem, DTMF circuit, LAN controller.

10. Explain in detail the following.

(i) Clock Oscillator (4)

(ii) Real time Clock (4)

(iii) Embedded system Memories (8)

(i) Clock Oscillator (4)

- Crystal oscillator
- Synchronize the timing operation
- Controls the timing in execution
- Types

  Crystal resonator, internal ceramic resonator, external IC based oscillator.

(ii) Real time Clock (4)

- Timer circuit suitably configured in the system clock
- Used for schedulers and real time programmers
- Example- generation of 0.5 μsec delay

(iii)Embedded system Memories

Explanation of all the types of memories.

Types of RAM & ROM-internal, external (4)

CACHE, EPROM, FLASH

Functions assigned to memories (4)

_____**********_____

UNIT II

DEVICES AND BUSES FOR DEVICES NETWORK

I/O devices – Device I/O types and examples – Synchronous ISO – Synchronous and asynchronous communications from serial devices – Examples of internal serial communication devices – UART and HDLC – Parallel port devices – Sophisticated interfacing features in devices/ports – Timer and counting devices – 12C, USB, CAN and advanced I/O serial high speed buses – ISA, PCI, PCI-X, CPCI and advanced buses

PART A

1. **Differentiate between control register and status register?**

Control register	status register
A register for bits which controls the action of a device	A register of bits which reflects the current status of the port buffer
For write operation only	For read operation only

2. **What is device encoder?**

A circuit to take the address bus signals as the input and generate a chip select signal for the port address selection.

3. **Differentiate parallel port and serial port?**

Parallel port	Serial port
A port for a read and write operation on multiple bits at an instance.	A port for an R/W operation with one bit at an instance and where each bit of the message is separated by constant time interval

4. **Define protocol**

   The way of transmitting messages on a network by using a software for adding the additional bits like starting bits, headers, addresses of source and destination, error control bits and ending bits.

5. **What is real time clock?**

   A clock that continuously generate interrupts at regular interval endlessly. An RTC interrupt ticks the other timer of the system.

6. **What is system clock?**

   A clock scaled to the processor clock and which always increments without stopping or resetting and generates interrupts at preset time intervals.

7. **What is timer overflow?**

   A state in which the no. of count inputs exceeded the last acquirable values and on reaching that state and interrupt can be generated.

8. **What is the use of PISO and SIPO**

PISO	SIPO
Used for serial bit reception in synchronous mode	Used for serial bit transmission in synchronous mode

9. **What is HDLC?**

   High level data level link control protocol for synchronous communication between primary and secondary. It is a bit oriented protocol.

10. **What is quasi bi-directional port?**

    A port with the dual advantage of using pull up circuits as per the voltage and current level required when interfacing it and using no full up circuit for a short period sufficient to drive a LSTTL circuit.

11. **What is meant by master slave communication?**
    A communication between two processors when one processor guides the transmission of the bits to a slave after receiving acknowledgement from the address slave.

12. **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.

13. 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.

14. 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

15. 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.

16. 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.

17. 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.

18. 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.

19. Why are SPCI parallel buses important?

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

20. 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.

21. What does UART contain?

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

22. 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.

23. Name the HDLC’s frame structure?

Flag	Address	Control	Data	FCS	Flag

24. 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

25. Name some control bit of timer?

- Timer Enable
- Timer start
- Up count Enable
- Timer Interrupt Enable

PART B

1. a) What are the types of I/O devices? Explain it with examples. (8)

Types of I/O devices (8)

1. Serial input

   Eg: audio input video input, scanner

2. Serial output:

   Eg: audio output video output re mote TV control

3. Serial UART input

   Eg: keyboard mouse, modem

4. Serial UART output:

   Eg: Modem printer character output on serial line

5. Parallel port single bit input:

   Eg: filling a liquid up to a fixed level, completion of revolution of a wheel, achieving preset pressure in a boiler

6. Parallel port single bit output

   Eg: Pulses to a external circuit, PWM output for DAC

7. Parallel port input:

   Eg: ADC input from liquid level measuring sensor, encoder inputs for bits for angular position of a rotating shaft

8. Parallel port output:

   Eg: multilane LCD display, printer or robot stepper motor coil driving output bits

b) List out the different states in a timer (8)

Timer States (8)

1) Reset State
2) Initial Load (Idle) State
3) Present State
4) Overflow State
5) Overrun State
6) Running (Active) or Stop (Blocked) state Finished (Done) state

7) Reset enabled/disabled State
8) Load enabled/disabled State
9) Auto Re-Load enabled/disabled State
10) Service Routine Execution enable/disable State

2. Discuss in detail about synchronous and asynchronous serial devices with examples and give its characteristics. (16)

Definition of Synchronous communication: (2)

When a byte or frame of the data is received or transmitted at constant time intervals with uniform phase differences

Characteristics of synchronous communication: (4)

1. Bytes (or frames) maintain a constant phase difference.
   - no permission for sending bytes at random time intervals.
   - No hand shaking during the communication.
2. Transmitting serially, the bits of all the bytes or frames.
   - The transmitter transmits clock rate information.
   - There are 2 separate lines for data bits and clock PISO and SIPO
   - Common line and clock information encoded by modulating clock with stream of bits

Five methods of encoding clock information into a stream of serial bits(2)

- Frequency modulation
- Mid frequency modulation
- Manchester coding
- Quadrature amplitude modulation
- Bi- phase coding

Definition Asynchronous communication: (2)

When a byte or frame of data is transmitted or received at variable time intervals

Characteristics of asynchronous communication are as follows (6)

1. Bytes (or frames) need not maintain a constant phase difference.
    - This mode facilitates handshaking between the serial transmitter and receiver port.

2. Though the clock must tick at a certain rate to transmit bits of a single byte (or frame) serially,
    - It is always implicit to the asynchronous data receiver.
    - Transmitter does not transmit along with the serial stream of bits.
    - The receiver clock does not maintain identical frequency and constant phase difference with transmitter clock.

3. a) Enumerate the formats of bits in a synchronous HDLC protocol based network device (8)

Format of Bits in a Synchronous HDLC Protocol-based Network Device (8)

S. No.	Bits at Port	Present Compulsorily or Optionally	Explanation
1.	Frame start and end Signaling flag bits	Compulsory	Flag bits at start and end are (01111110)
2.	Address bits for destination	Compulsory	8 bits in Standard format and 16 bits in extended format
3a.	Control bits Case 1: Information Frame	Compulsory as per case 1 or 2 or 1	First bit 0, next 3-bits N(S), next bit P/F and last 3-bits N(R) in standard format &N(R) and N(S) = 7 bits each in extended format
3b.	Control bits Case 2: Supervisory Frame	--------	First two bits (10), next 2-bits# RR or RNR or REJ or SREJ, next bit PIP and last 3-bits N(R) in standard format. N(R) and N(S) = 7-bits each in extended format
3c.	Control bits Case 3: Un-numbered	--------	First two bits (11), next 2-bits AM, next bit P/F and last 3-bit remaining bits for M. [8-bits are immaterial after M bits in extended format]

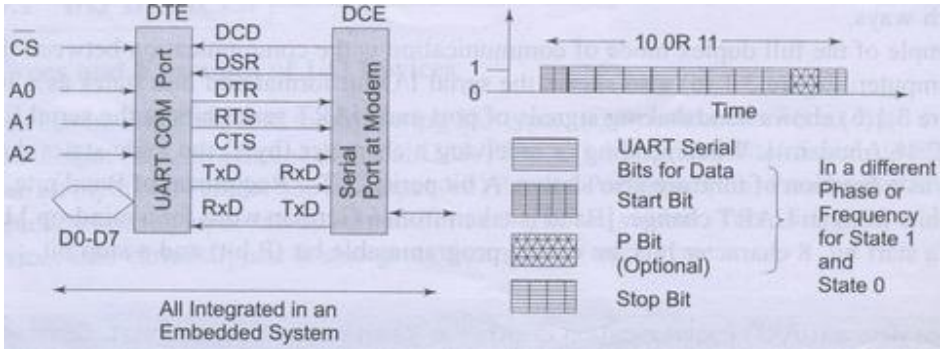
	Frame		
4.	Data bits	Compulsory	m frame bits transmit such that each bit is at the line for time Δ T or, each frame is at the line for time mΔ T.
5.	FCS (Frame Check Sequence) bits	Compulsory	16 bits in standard format and 32 in extended format
6.	Frame End flag bits	Compulsory	Flag bits at end are also (01111110)

3. b) What is UART? Discuss the handshaking process in UART serial port. (8)

Definition of UART: (2)

A standard asynchronous serial input and output port for serial bits.In micro controllers a start bit precedes the 8 bit message and a stop bit succeed the message

Figure: The handshaking signals used in UART serial port (6)

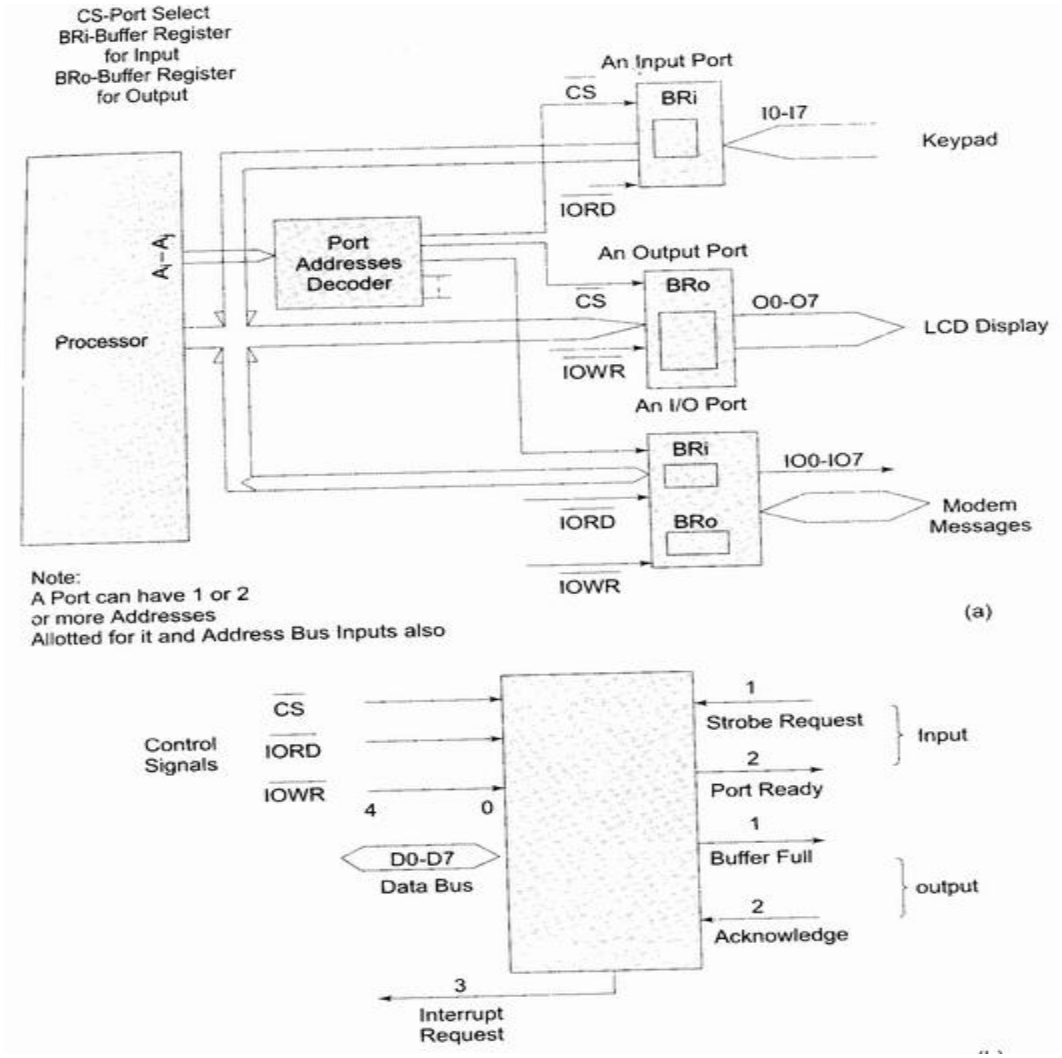


A bit period δt = reciprocal of baud rate, the rate at which the bit from an UART change

UART bits consists of a start bit, 8 character bits (compulsory), a option programmable bit and a stop bit.

4. a) Explain in detail about parallel port devices with neat diagram? (8)

figure: Parallel input Port, output port and a bi-directional port for connecting the device (4)



explanation: (4)

it can be I0 to I7 input bits from a keypad input controller

O0 to O7 are the serial output bits to LCD display output controller

Br_i and Br_o are buffers at the input port and output port

Each port connects to the address bus signals, A_i, and A_j through a port address decoder

IORD and IOWR are additional control signals for a port read and write.

4, b) List out considerations when interfacing a device port (8)

Characteristics taken into consideration when interfacing a device-port (8)

1. A port device may have multi-byte data input buffer(s) and data output buffer(s).
2. A port may have a DDR. This is advantage since each bit of the port is programmable. It can be set as input or output DDR programs the port bits.
3. Port LSTTL driving capability and port loading capability are important characteristics. A port may be an OD (open drain) port. It has zero driving capability. If the given port has the OD gates, appropriate pull-up resistance or transistor is connected to each gate to have driving11 capability. .
4. If the given port is quasi bi-directional then the port has limited driving capability for a period of one or few clock cycles and for one or a few LSTTL gates only. When this device port connects to more than one LSTTL then an appropriate pull-up circuit is connected to each gate.
5. There may be multiple or alternate functionality in the port; pins,.
6. A port may have provision fort multiplexed output to connect to multiple systems or units
7. A port may have provision for demultiplexed inputs from the multiple systems or units

5. **Briefly explain the sophisticated interfacing features in device ports (16)**

Features (16)

1. In-built Schmittt circuit at the port is conditioning of the signal by noise-elimination.
   - Such a device is extremely useful in transceivers for repeating systems. These are used in long distance communication.
2. When a port device is waiting for instructions, power management can be done at the gate in the device.
   - Data Gate-like circuit at a device is reduced power dissipation when the port device is operated at fast speeds,
3. Port interfaces used to be either open drain CMOSs or TTLs or RS232Cs.
   - Examples are HSTL, SSTL.
   - HSTL is used for high-speed operations,
   - SSTL is used when the buses are to be isolated from relatively large stubs.
4. A device connects to a system bus and also to an I/O bus

5. An I/O device may consist of multiple gigabit (622 Mbps to 3.125 Gbps) transceiver(s). Rocker I/O TM serial 3.125 Gbps transreceivers are the examples of circuits that provide the support circuitry for this rate.
6. A device for I/O may integrate a SerDes (serialization and De-serialization) subunit. SerDes is a standard subunit in a device.
   - The great advantage of the SerDes unit is that these operations are fast.

7. Multiple I/O standards have been developed for I/O devices. Advantages of multiple standard support devices are obvious.

8. I/O device may integrate a digital Physical Coding Sub layer (PCS). Analog audio and video signals can then be pulse code modulated (PCM) at the sub layer.
   - The advantage of an in-built PCS at a port device is that there is no need of external PCM coding.

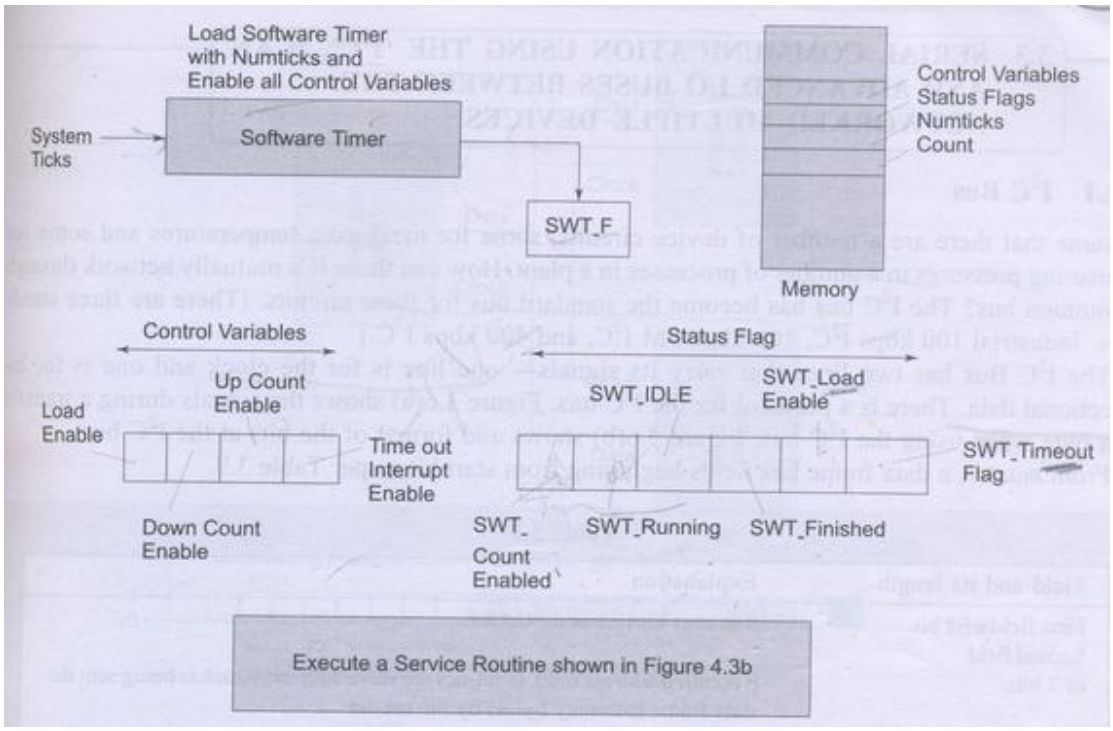
9. A device for I/O may integrate an analog Physical Media Attachment (PMA) unit for connecting direct inputs and outputs of voice, music, video and images.
   - advantage of an in-built PMA is that the device directly connects to the physical media.

6. Explain hardware timer and software timer (16)

Important points to be covered: (5)

- It is a virtual timing device.
- The system clock or any other hardware-timing device generates interrupts at periodic intervals; this interrupt is as per the count value set. Now the interrupt becomes a clock input to a software timer.
- SW control bits are set as per the application.
- These can be up to nine types of control bits.
- There is no hardware input or output in SWT.
- It includes control bits and status flag.
- Physical limit (1, 2 or 3 or 4 for the number of hardware timers in a system).
- SWT's can be unlimited in number.

Figure: software timer (3)

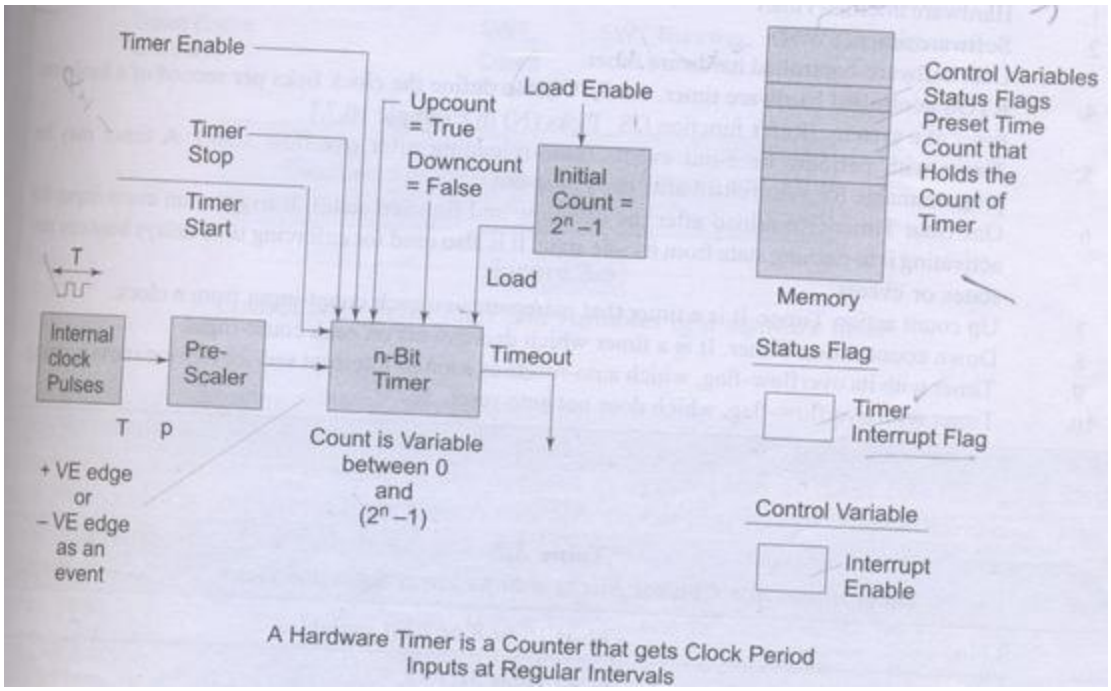


Hardware timer:

Important points to be covered: (5)

- At least one hardware timer device is a must in a system
- It is used a system clock.
- It gets the input from a clock out signal from the processor
- Control bits can be of 9 types
    - Timer enable
    - Timer start
    - Timer stop
    - Prescaling bit
    - Up count enable
    - Down count enable
    - Load enable
    - Timer interrupt enable
    - Timer interrupt enable (reaches count= 0)

Figure: hardware timer(3)



7. a) Features of on chip serial devices in micro controller (8)

Features (8)

1. Synchronous Serial Port (Half or Full Duplex)
2. Asynchronous UART Port (Half or Full Duplex)
3. Programmability for 10 as well as 11 bits per byte from UART
4. Separate un-multiplexed Port Pins for Synchronous No and UART Serial ports
5. Synchronous Serial Port as a Master or Slave
6. UART Serial Port as a Master or Slave define by Software programming for P bit
7. Synchronous Serial Port Registers
8. UART Serial Port Registers
9. Uses Internal Timer or Uses Separate programmable BAUD rate generator.

7. b) Enumerate the uses of timer devices, explain each with its application (8)

Uses of timer device(8)

1. Real Time Clock Ticks (System Heart Beats).
2. Initiating an event after a preset delay time.
3. Initiating an event after a comparison(s) between the preset time(s) with counted value(s).
4. Capturing the count value at the timer on an event.
5. Finding the time interval between two events.
6. Wait for a message from a queue or mailbox or semaphore for a preset time when using RTOS.
7. Watchdog timer.
8. Baud or Bit Rate Control for serial communication on a line or network.
9. Input pulse counting when using a timer
10. Scheduling of various tasks.
11. Time slicing of various tasks
12. Time division multiplexing (TDM)

8) a)Explain the concept in I^2C Bus? (10)

definition (2)

I^2C is a serial bus for interconnecting ICS.It has a start bit& stop bit like in UART.It has Seven fields for start,7 bit address,defing byte as acknowledging byte, data byte,Nack and end.

Important points to be covered (4)

I^2C bus has two lines that carry its Signal.

One line is for clock,another one for bidirectional data.

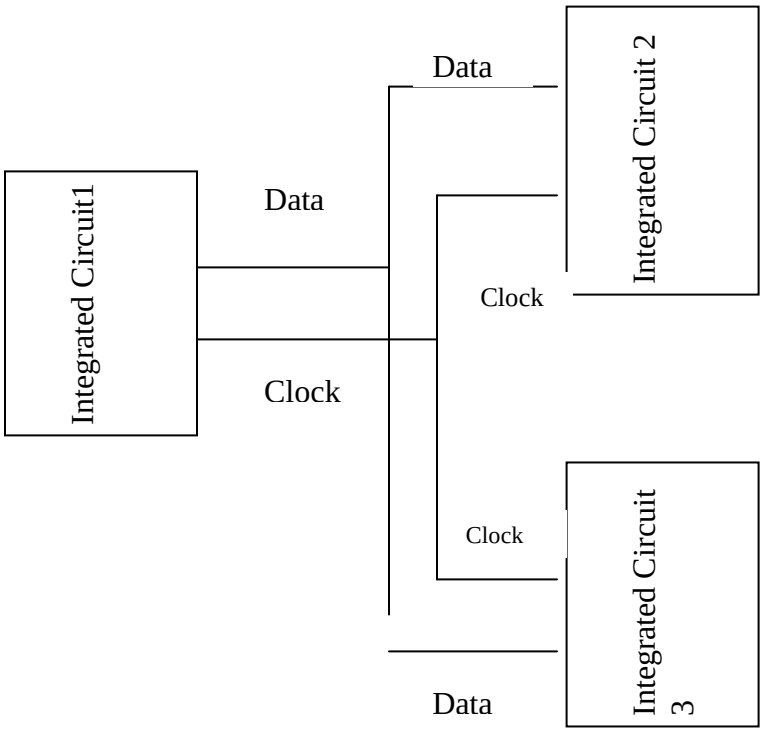
Data frame has fields begining from start bit

Fields and its length	**Description**
First field 1 bit	It is start bit like in UART
Second fields 7bit	It is Address field,it defines the slave address
Third fields 1 bit	It is control bit field ,it defines whether

		read (or) write cycle
Fourth fields	1control bit	It defines whether the present data is an Ack
Fifth fileds	8bit	It is for IC device data byte
Sixth filed	1bit	It is a bit neg-Ack(NACK)
Seventh	1 bit	It is a stop bit like UART

Diagrams **(4)**

Signals during transfer of a byte when using the I^2C bus



Format of bits at the I^2C bus

Start Bit	Slave Address	R/W	ACK	Data bits		Stop Bit

Disadvantages:

- Time taken by the Algorithm in the master hardware that analses the bits through I^2C .
- Slave Hardware does not provide for the hardware that supports it.

8) b)Explain the USB bus? (6)

DEFINITION: (2)

It 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 n/wing the i/o devices like scanner in a comp.system.

Important points to be covered (2)

- It provides a fast (up to 12 mbps) and low speed (up to 1.5 mbps)
- Serial transmission and reception b/w host& serial devices like Scanner,Keyboard,printer
- There are two standards
- USB 1.1 (a low speed 1.5mbps 3meter channel along with a high speed 1.2 mbps 25 meter)
- USB 2.0.(High speed 480 mbps 25 meter channel)

Features (2)

- Device can be attached,configured is used reset,reconfigured and used,detached and reattached
- It can be either bus powered
- USB bus cable has 4 wires one for +5,2->Twisted pairs,1->Ground
- Data transfer is 4 types i) Controlled data transfer ii) Bulk data transfer iii)Interrupt driven data transfer iv) Iso synchronous transfer
- USB is a polled bus
- It supports three types of pipes i) Streams ii)Default control iii)message

9) a)Describe the CAN bus? (10)

It is a Serial bus for interconnecting a central control n/w.It is mostly used in Automobiles.

(2)

It has fields for bus arbitration bits,control bits for Address and data length,data bits CRC check bits,Ackowledgement bits and ending bits.

- It has a serial line,which is bidirectional. **(4)**
- Maximum data Transfer rate in 1mbps
- Twisted pair connection to each node
- Runs up to max.length of 40m.
- Line is at logic'1' in its idle state called "recessive state"
- Line is at logic'0' in its idle state called "dominant state"
- Data frames always start with '1' and always end with seven '0's
- It usually interconnects to a CAN controller between line and host at the node
- There is an Arbitration methods called CSMA/Amp **(4)**

Field and its	length	Description
First field	12 bits	Arbitration fileds
Secondfield	6 bits	Control fields
Third field	0 to 64 bits	Data length code
Fourth field	16 bits	CRC
Fifth field	2 bits	First bit "Ack slot",Second bit Ack delimiter
Sixth fields	7 bits	End of the frame

9) b)Explain Advanced Serial High Speed bus? (6)

An embedded System may need to connect multi gigabits per second(Gbps),Transceiver(Transmit and receive) serial interfaces **(3)**

Eg **(3)**

IEEE 802.3-2000
XAUI (10 gigabit Attachment unit)
ATM oc-12/46

10) a)Explain the ISA bus? 8

Definition:

ISA bus (used in IBM standard Architecture) connects only to card that has 8086,186,286 processor and in which the processor addressing & IBM pc Architecture addressing limitation & interrupt vector address assignments **(2)**

Features:

- Memory access can be in two ranges 640 kb to 1 mb,15mb to 16 mb **(6)**
- Only 1024 I/O port address are available
- Address allocated are hex 000-00f for DMA chip 8237
- Hex 020-021 address allocated are for programmable interrupt controller 8255
- Reserved address from peripherals are hex 220-24F,278-27F
- Display monitors ports are within hex 380-38F and 3D0-3DF

10) b) Explain the PCI bus? 8

- It provides a superior throughput than EISA. **(4)**
- It is almost platform Independent
- Its clock rate is nearest to the sub multiple of system clock
- PCI provides 3 types of Synchronous parallel interfaces
- Its two version 32/33 mhz,64/66 mhz
- Two super speed version of pci have been introduced
- PCI super V2.3 264/528 Mpbs 3.3v(on a 64 bit bus),132/264(on 32 bits)
- PCI bus has 32 bit data bus extensibility to 64 bits
- Its Synchronous/Asynchronous throughput is up to 132/528 mb/s

- PCI card has 16 mb flash Rom
- PCI bus attachement and detachment of the system peripheral
- Three identification number by which device identifies its addresss space i) I/o port ii) memory location iii) configuration reg

64 bytes at the standart device independent configuration reg in a PCI device

(4)

0x30	**EXP ROM**		**Reserved**			**IRQ IRQ LINE PIN HT MAX GNT**				
0x20	BA4		BA5			CBCISP		SSVID	SSDID	
0x10	BA0		BA1			BA2		BA3		
0x00	VID	DID	CR	SR	RID	CC	CL	LT	HT	BIST

VID:Vendor ID DID:Device ID RID:Revision ID CR:Common Register
CC: Class code SR:Staus Register CL : Cache line LT:Latency Timer
BIST:Base input Tick HT:Header Type BA-base address
CBCISB:Card Base CIS pointer SS:sub system ExpRom:Expansion Rom

UNIT-III

PROGRAMMING CONCEPTS AND EMBEDDED PROGRAMMING IN C,C++

Programming in Assembly Language (ALP) vs high level language – C program elements – Macros and functions – Use of pointers – NULL pointers – Use of function calls – Multiple function calls in a cyclic order in the main function pointers – Function queues and interrupt service routines – Queue pointers – Concepts of embedded programming in C++ – Objected oriented programming – Embedded programming in C++ – C program compilers – Cross compiler – Optimization of memory codes

PART – A

1. Advantages of high level language over assembly language: **(2)**

- gives a precise control of the processor internal devices
- full use of processor specific features and its instructions set as well as addressing mode
- machine codes are compact i.e., code does not contain the declaration of conditions, rules and data types. so system needs a small memory

2. Executable file: **(2)**

- File that is used to put the initial data, tables, strings, vectors and source code in the ROM.

3. Some of include files and its purpose in embedded 'C' system: **(2)**

- Code files – files for the codes already available
- Constant data files – files for the code and may have the extension '.const'
- String data files – files for the code and may have the extension '.str' or '.strings' or '.txt'

4. Difference between a header file over a constant file: **(2)**

- The header files are well tested and debugged modules
- The header files provide access to standard libraries
- The header files can include several text files or 'C' files

5. Reentrant function: **(2)**

- All the argument passes the values and none of the argument is a pointer whenever a calling function calls that function.

6. Recursive function is not used in embedded system because: **(2)**

- Because of memory constraints.
- For example STACK goes after each recursive call and it may choke the memory space availability.

7. Example for infinite loop: **(2)**

- #define false 0
  #define true 1
  While(true)
  {
  /*code that repeatedly execute until the condition is false or 0*/
  }

8. Importance of the following declarations: static, volatile and interrupt: **(2)**

- Static: It allows main ( ) to be called with out having to instantiate a particular instance of the program.
- Volatile: It tells the compiler that the variable modified by volatile can be changed unexpectedly by other parts of a program.
- Interrupt: During the normal program execution if request for some other program execution arises the processor suspends its normal operation and starts executing the requests.

9. Declare NULL pointers in embedded 'C': **(2)**

- # define NULL (void*) 0x0000
  Instead of 0x0000 addresses use any address

10. Need for cross compiler: **(2)**

- In host itself develop the machine codes for target system.

11.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.

12. What is scheduling policy?

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

13. Define hyper period?

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

14. What is schedulability?

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

15. What are the types of scheduling?

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

16. 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.

17. Define round robin scheduling?

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

18. What is scheduling overhead?

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

19. What is meant by context switching?

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

20. Define priority scheduling?

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

21. What is rate monotonic scheduling?

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

22. What is critical instant?

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

23. 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.

24. 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.

PART B

1. Discuss in detail about the advantages of Assembly Language and High Level language in embedded 'C'? (16)

1. <u>Advantage of Assembly Language</u> (8)

- gives a precise control of the processor internal devices
- full use of processor specific features and its instructions set as well as addressing mode
- machine codes are compact i.e., code does not contain the declaration of conditions, rules and data types. so system needs a small memory
- Excess memory needed does not depend on the programmer data type declaration and rule declarations. It is also not the compile specific and library functions specific
- device driver codes may need only a few assembly instructions
- In assembly code for the small embedded system like timer device for microwave oven and washing machine the device driver code is compact and precise

<u>Advantage of High Level Language embedded 'C'</u> (8)

- Development cycle is short for complex systems due to the use of functions(procedures), standard library functions and modular programming approach
- Application programs bases on the Software engineering principles i.e.,
    i) Repetitive coding redundant
    ii) Use of the Standard Library functions
    iii) Modular programming approach
    iv) Bottom up design
    v) Top down design
- Data type declarations provide programming very easy
- Type checking makes program less prone to error

- Control structures and Conditional statements make the program-flow path design tasks simple
- Portability of non processor specific codes exists. Therefore, when hardware changes, only the modules for the device drivers, device management and initial boot up record data need modifications.
- Additional advantage of using embedded 'C' as a high level language, inserting the assembly language codes in between the embedded 'C' coding. This is called in – line assembly.

2. Explain the following programming elements with suitable syntax and examples.

a) Include directories (8)
b) Source files (2)
c) Configuration files (2)
d) Preprocessor directives (4)

a) Include directories: (8)

-Any 'C' program first includes the header files.

- Header files contain any specific function codes which was debugged.

Ex: # include "vxWorks.h" - it contains Vx Works functions

include "taskLib.h" - it contains Multitasking functions library.

- Include is preprocessor directive to include the contents (code or data) of a File.

i) Including code files (having '.c' extension)

ii) Including constant files (having '.const' extension)

iii) Including string data files (having '.str' or '.txt' or '.strings' extension)

iv) Including initial data files (having '.init' or '.data' extension)

v) Including basic variable files (having '.bss' extension)

vi) Including Header files (having '.h' extension)

Difference between a header file over a text file:

- The header files are well tested and debugged modules
- The header files provide access to standard libraries
- The header files can include several text files or 'C' files

b. Source files: (2)

- Source files are program files for the functions of application software.
- Source files need to be compiled
- A source file will also possess the preprocessor directives of the application and have the first function from where the processing will start. This function is called main function.

c. Configuration files: **(2)**

- Configuration files are the files for the configuration of the system.

  Eg: "#include "os_cfg.h" – It will include operation system configuration header file

d. Preprocessor directives: **(4)**

- Preprocessor Global Variable

  Eg: #define volatile Boolean IntrEnable
  Here IntrEnable is a global variable of Boolean datatype and is volatile

- Preprocessor Constants

  Eg: #define false 0
  Here constant name is false and assumed is equal to zero.

3. (i.) Define the following terms:

a) Macros **(3)**
b) Functions **(3)**
c) Stacks **(2)**

(ii) Explain the usage of FIPO Queues for Flow control on a Network.(8)

i) a) Macros: (3)

- A macro is a collection of codes that is defined in a program by name.
- Where ever the macro names occur in the main function that name replace by the codes.
- The codes for a macro compiled for every place it occurs

b) Functions: (3)

Function also a collection of codes.

- Here the values passed by the calling program through its arguments.
- Also returns a data object when it is not declared as void.
- The codes for a function compiled once only.

c) Stack:

(2)

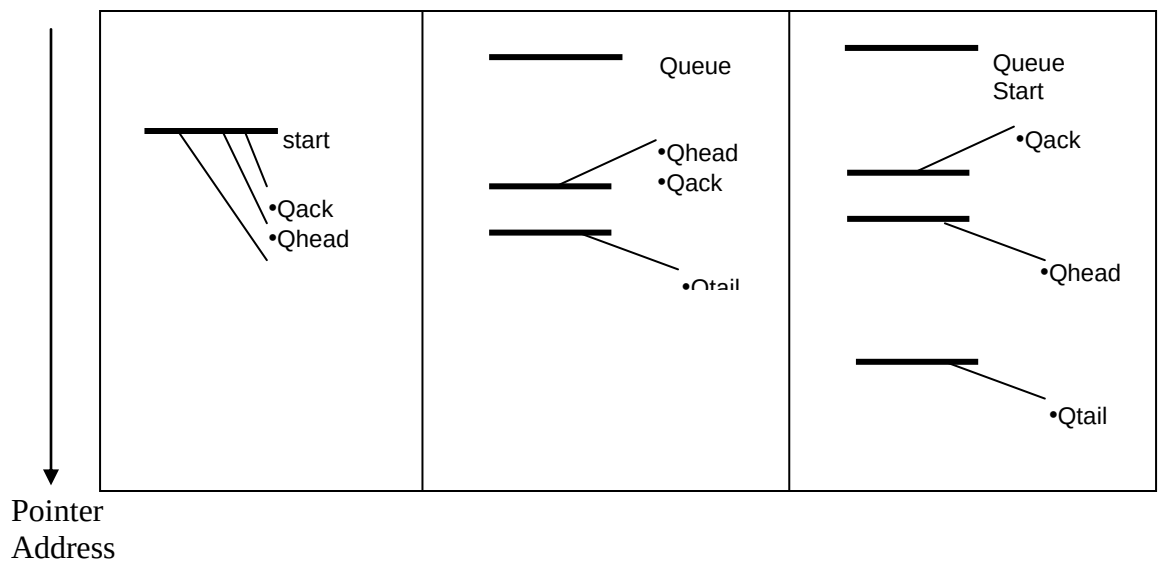
- It is a structure with a series of elements with its last element waiting for an operation.
  - An operation can be done only in the last in first out (LIFO) mode.
  - An element can be pushed only at the top in the series of elements still waiting for an operation.

Eg: Pushing of variables on interrupt or call to another function.

ii) Use of the FIPO (First-In Provisionally – Out) Queues for Flow Control on a Network (8)

- A commonly used network transport protocol is 'Go back to N'. It is used in case of a point-to-point network.
- Bytes transmit from the network driver (transmitter) and queue up to a certain limiting number or up to the occurrence of a time-out, whichever is earlier.

Time as Transmission Through a Socket Progresses



1. front (*QHEAD) equals back (*QTAIL) as well as tempfront (*QACK) at the beginning of the transmission.
2. When there is an acknowledgement , front (*QHEAD) resets and equals tempfront (*QACK).
3. The transmission starts from the tempfront(*QACK) again.
4. There is a limiting maximum time interval difference between transmission from tempfront (*QACK) and after that time if tempfront(*QACK) is not equal to front (*QHEAD) then front(*QHEAD) resets and equals tempfront (*QACK) again.

4. Explain in detail about the implementation and functions of QUEUE in embedded 'C' language (16)

Queue:

- Types of Queue – a) Queue b) Circular queue and its definition (6)

- A queue can be restricted to 256 elements in a system that has a small memory.
- A design concept that can be very useful in an embedded networking system is used for coding.
- It differs from a conventional application coding for a queue as follows.

- There are five flags are used.
    i) Qerrorflag
    ii) header flag
    iii) trailing flag
    iv) CirQuFlag
    v) PolyQuFlag

- There are four Boolean variables
    i) headerEnable
    ii) trailingEnable
    iii) CirQuEnable
    iv) PolyQuEnable

- There are four unsigned short variables
    i) headerNumbyte
    ii) trailingNumbyte
    iii) CirQuNumbyte
    iv) PolyQuBlockNum

Use of the Queues for Implementing the Protocol for a Network: (5)

- Networking applications need the specialized formations of a Queue.

- diagram and explain

Queuing of Functions on Interrupt: (5)

- diagram and explain

- A source with a smaller deadline for its service can be serviced without missing its service
- On a return from a service routine, the operation function 'operationPortA( )' Gets the function pointers from the queue and then executes the pointed function.

5. What are the usage of Data types and Data Structures in embedded 'C' language? (16)

Usage of Datatypes: (4)

- Whenever a data is named , it will have the address allocated at the memory.
- The number of addresses allocated depends upon the data type.
- 'C' allows the following primitive data types.

char	- 8 bit for character
byte	- 8 bit for character
unsigned short	- 16 bit
short	- 16bit
unsigned int	- 32bit
int	- 32bit
long double	- 64bit
float	- 32bit
double	- 64bit

- A datatype appropriate for the hardware used.

Declaration of a datatypes:

```
typedef unsigned character portAdata
#define Pbyte portAdata Pbyte = 0xF1
```

Usage of data Structures: (4)

- A data structure is a way of organizing large amounts of data.
- A data element can be identified and accessed with the help of a few pointers and /or indices and/or functions.

Data structure	Definition and when used	Example of its use
Queue	It is a structure with a series of elements with the first element waiting for an operation. An operation can be done only in First In First Out(FIFO)	1. Print buffer. 2. Frames on a network.
Stack	It is a structure with a series of elements with its last element waiting for an operation. An operation can be done only in Last In First Out(LIFO)	1. Pushing on variable on interrupt to another function 2. Retrieving the pushed data onto a stack.
Array (One dimensional array)	It is a structure with a series of elements with each element accessible by an identifier name and an index.	1. int a[12] here we have store the element a[0] ….a[12] and accessing use this(a[0..12]) index
Multidimensional array	It is a structure with a series of elements with each having another sub-series of elements. Here also accessing is done by identifier name and two or more indices.	1. Handling of Matrix. Like 2X2,3X3 matrix.
List	Each element has a pointer to its next element. Only the first element is identifiable and it is done by list-top pointer(Header)	1. A series of tasks which are active Each task has pointer for the next task. 2. A menu that point to a submenu.
Tree	There is a root element. It has tow or more branches each having a daughter element. Each daughter element has two or more daughter elements. The last one does not have daughters. Only the root element is identifiable and it is done by the treetop pointer (header).	1. An example is a directory. It has number of file-folders. Each file-folders has a number of other file folders and so on In the end is a file.

- Important data structures are stack, one dimensional array, queue, circular queue, pipe, a table, look up table, hash table and list.

6. Define the following in embedded 'C' language.

a) Use of Conditions, Loops and Infinite Loops (8)

b) Usage of Pointers and Null Pointers

(8)

. (i) Use of Conditions, Loop and Infinite Loops: (8)

- If a defined condition is fulfilled, the statements within the curly braces after the condition are executed; otherwise the program proceeds to the next statement or to the next set of statements.
- Set of statements is repeated in a loop. In array, the index changes and the same set is repeated for another element of the array.
- Infinite loop is a feature in embedded system programming.

Example for Infinite Loop:

(a).
```
#define false 0
#define true 1
 void main(void)
  {
  while (true) /*the loop is repeated until the condition is true.*/
   {…….}
```

(b) /*preemptive schedule*/
```
#define false 0
# define true 1
void main(void)
{
rtos.run( );
}
void task1(…..){
while(true){
if(flags1){…..,};flag1=0;
message1( );
}
}
void task2(…..){
while(true){
if(flags2){…..,};flag2=0;
message2( );
}
}
void taskN(…..){
```

```
while(true){
if(flagsN){…..,};flagN=0;
messageN( );
}
}
```

(ii) Use of Pointers and NULL Pointers: (8)

- Declare the byte datatype:
  Ex: unsigned byte *portA
  Here * means 'the content at' .This declaration means that there is a pointer and an unsigned byte for portA.

- Declaration of a function:
  Ex: void *portAdata
  Here void means undefined data type for portAdata.

- Declaration of a preprocessor directives:
  Ex: #define portA(volatile unsigned byte*)0x1000
  Here pointer can be assigned a constant fixed address.

- Declaration of indirection operator:
  Ex: unsigned byte *portA=&portAdata
  Here & means 'at the address of' and the -positive number of 8 bits pointed by portA is replaced by the byte at the address of portAdata.

- Declaration of NULL pointer:
  Ex:#define NULL(void*)0x0000

7. i) Define the following in embedded 'C' language.
a) Usage of Modifier (12)
ii) What are the advantages of building ISR queue? (4)

Usage of Modifier: (12)

Case i: Modifier 'auto' or No modifier, if outside a function block, means that there is ROM allocation for the variable by the locator if it is initialized.

Case ii: Modifier 'auto' or No modifier, if inside a function block, means that there is ROM allocation for the variable by the locator if it is initialized. But here no RAM allocation by the locator

Case iii: Modifier 'unsigned' is modifier for a short or int or long data type.

Case iv: Modifier 'static' declaration is inside a function block. Static declaration is

a directive to the compiler that the variable should be accessible outside the function block also and there is to be a reserved memory space for it.

Case v: Modifier 'static' declaration is outside a function bloc. It is not usable outside the class in which declared or outside the module in which declared.

Case vi: Modifier const declaration is outside a function block. It must be initialized by a program.

Case vii: Modifier register declaration is inside a function block. It must be initialized by a program.

Case viii: Modifier interrupt. It directs the compiler to save all processor registers on entry to the function codes and restore them on return from that function.

Case ix: Modifier extern. It directs the compiler to look for the data type declaration or the function in a module other that the one currently in use.

Case x: Modifier volatile outside a function block is a warning to the compiler that an event can change its value or that its change represents an event.

Case xi: Modifier volatile static declaration is inside a function block. The static declaration is for the directive to the compile that the variable should be accessible outside the function block.

ii) Advantage of building ISR queue: (4)

- It reduces significantly the ISR latency periods.
- Each device ISR is therefore able to execute within its stipulated deadline.

8. (i)Discuss about the Usage of Function Calls in embedded 'C' Language. (8)
(ii) Discuss about the multiple function calls in Cyclic order. (8)

(i) Use of function calls: (8)

Declaring a function:

- Each variable has to have a declaration, each function must be declared.
  Ex: int run (int indexRTCSWT, unsigned int maxlength);

Defining the statements in the function:

- Each variable has to be given the contents or value each function must have its statements.

Ex: int RTCSWT::run (int indexRTCSWT,unsigned int maxlength)
{…….}

Call to a function:

- There is a call on fulfilling a condition
  Ex: if (delay_F= = true && WTDelayIEnable= =true)
  ISR_Delay( );

Passing the values:

- The values are copied into the argument of the functions.
- When the function is executed,it does not change a variables value at the called program.

Reentrant function:

- All the arguments pass the values and none of the argument is a pointer whenever a calling function calls that function.
- When an operation is not atomic, that function should not operate on any variable, which is declared outside the function or which an interrupt service routine uses or which is a global variable but passed by reference and not passed by value as an argument into the function.
- That function does not call any other, function that is not itself reentrant.

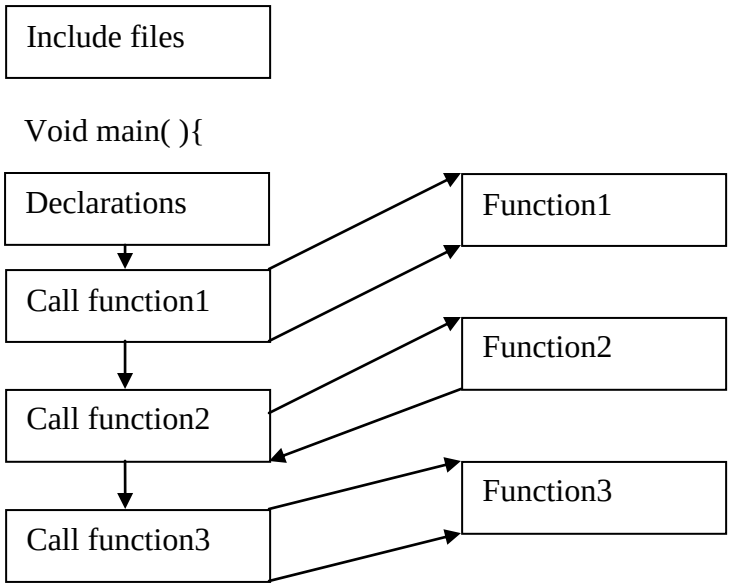
Passing the references:

- When an argument value to a function passes through a pointer, that function can change this value.

(ii) Multiple function calls in cyclic order in the main (8)

- One of the most common methods is for the multiple function calls to be made in a cyclic order in an infinite loop of the main.

Programming Model



9. Explain in detail about LISTS in embedded 'C' language. (16)

List:

Definition: (2)

Each element has a pointer to its next element. Only the first element is identifiable and it is done by list-top pointer (Header).No other element is identifiable and hence is not accessible directly.

List differs from array, ordered list, queue: (3)

- In a list, each element must include an item as well as a pointer.
- In an array, the memory allocation is as per the index assigned to an element.
- In ordered list, all the elements rearrange according to a priority parameter in the ordered list on creation, on any new insertion or on any deletion.
- In a queue, which is accessible and is readable as FIFO (First In First Out) only.

Use of a List of Active Device Drivers (Software Timer):

Draw the diagram of programming model of list. (3)

Each element at an instance stores the following five programming variables: (2)

(i) Priority (ii) ItemID (iii) State (=active or inactive) (iv) C, counts that are remaining for finish. (v) Pointer for the next element,*pNext.

Use of a List of Tasks in a Ready list: (3)

Creation and deletion from the ordered list of ready tasks is done as follows by the OS.

1. The OS creates an ordered list of initiated (ready)-state tasks. These task are the ones that have been flagged to be prepared for execution by the CPU in an order.
2. When the OS receives a message for initiating a task into the ready state, the task from the idle state is inserted into a list of initiated (ready) tasks.
3. The codes in each listed task are then executed as directed by the OS.
4. A task, which finishes is deleted from the listed tasks on a direction from the OS.

Draw the diagram of programming model for multitasking operations. (3)

Each element at an instance stores the following five programming variables:

(i) Priority (ii) Id for the task. (iii) State (=running, ready or idle) (iv) Codes for the task that has to be executed when readied. (v) Pointer for the next list-element,*pNext.

10. i) Discuss about a)compiler and b) Cross compiler in embedded 'C' language (4)
ii) What are the techniques used for optimizing the memory. (12)

. (i) (a) Compiler : (2)

- It does the development and design and also the testing and debugging of a host.

(b) Cross Compiler: (2)

- It runs on a host, but it develops the machine codes for a targeted system.

Ex: GNU Compiler

- It is configurable both as host compiler as well as cross compiler.
- It supports 80x86, window 95/NT, 80x86 Red Hat Linux and several other platforms.

(ii) Optimization of memory (12)

- Use declaration of a variable as unsigned byte (0 to 255), because byte arithmetic takes less time than integer arithmetic.
- Avoid use of library function if a simpler coding is possible.
- - Use the assembly codes for simple functions like configuring the device control register, port addresses and bit manipulations if the instruction set is clearly understood.

  - Use assembly code for the atomic operations for increment an addition.

  -Use modifier 'register' for a frequently used variable.

- Use inline modifiers for all frequently used small set of codes in the function if the ROM is available in the system.
- Use global variables if shared data problems are tackled and use static variables in case it needs saving frequently on the stack.
- Whenever feasible combine two functions of more of less similar codes.
- Reduce use of frequent function calls and nested calls and thus reduce the time and RAM memory needed for the stacks respectively.
- Use if feasible, alternatives to the switch statement with a table of pointer to the functions.
- To free the RAM used by a set of statements, use the delete function and destructor functions.
- For using C++, use the classes without multiple inheritances, with template, with runtime identification and with throwable exceptions.

Unit IV

Real Time Operating Systems- Part – I

Definitions of process, tasks and threads – Clear cut distinction between functions – ISRS and tasks by their characteristics – Operating system services – Goals – Structures – Kernel – Process management – Memory management – Device management – File system organization and implementation – I/O subsystems – Interrupt routines handling in RTOS.

REAL TIME OPERATING SYSTEMS : RTOS task scheduling models – Handling of task scheduling and latency and deadlines as performance metrics – Co-operative round robin scheduling – Cyclic scheduling with time slicing (Rate monotonics cooperative scheduling) – Preemptive scheduling model strategy by a scheduler – Critical section service by a preemptive scheduler – Fixed (Static) real time scheduling of tasks.

INTER PROCESS COMMUNICATION AND SYNCHRONISATION: Shared Data problem – Use of semaphore(s) – Priority inversion problem and deadlock situations – Inter process communications using signals – Semaphore flag or mutex as resource key – Message queues – Mailboxes – Pipes – Virtual (Logical) sockets – Remote Procedure Calls (RPCs).

PART-A

1. **What should be the goal of an OS?**
   - Providing a common set of interfaces.
   - Orderly access and control.
   - Perfection, Correctness.
   - Interoperability, Portability.

2. **Why does an OS functions provide two modes, user mode and supervisory mode?**
   - OS is the middle layer between application software and system hardware.

- Application software operates in the user mode.
- System hardware operates in the supervisory mode.

3. **What is meant by RTOS?**
   - OS for response time controlled and event controlled processes.
   - OS for Embedded System as these have real time programming issues to solve.

4. **What are the functions of RTOS?**

   - RT task scheduling.
   - Interrupt-latency control.
   - Use of timers.
   - Use of system clocks.

5. **List three ways in which an RTOS handle the ISRs in a multitasking environment.**

   - An ISR servicing directly informing the RTOS.
   - Kernel intercepting the call and calling the corresponding ISR and task.
   - Kernel calling the ISR for the parameters (IPCs) by context switch and later executing the task corresponding to the source.

6. **When is an RTOS necessary and when is it not necessary in the Embedded Systems?**

      - May not be necessary in a small-scale embedded system.
      - Necessary when scheduling of multiple processes, ISRs and devices is important.
      - Must to monitor the response time controlled processes and event controlled processes.

7. **What are the three common model strategies that a scheduler may adopt?**

   - Control flow strategy.
   - Data flow strategy.
   - Control-Data flow strategies.

8. **Define Process, Task and Thread.**

   Process:

      - Computational unit that processes on a CPU under the control of a scheduling kernel of an OS.

   Task:

      - Set of a computation that processes on a CPU under the control of scheduling kernel.

   Thread:

      - Lightweight sub-process or process in an application program.

- Controlled by the OS Kernel.

9. Compare Scheduling strategies, Round Robin and Real time Scheduling.

CHARACTERISTICS	ROUND ROBIN	REAL TIME
Worst response time for task code	Total of execution time for all task code.	Zero.
Stability of response when the code changes	Good for interrupt routines. Poor for task code.	Very good.

10. Define preempting scheduling and time slicing scheduling.

Preempting Scheduling:

- A higher priority task is forced or preempted to block by the scheduler to at a higher priority task run.

Time slicing Scheduling:

- Each task is allotted a time slice after which it is blocked and waits for its turn on the next cycle.

11**. 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.

12. 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.

13. 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.

14. What is Priority inversion?

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

15. 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.

16. 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.

17. Define Mailbox and Pipe.

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

18. Define Socket.

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

19. 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.

20. List the functions of a kernel.

- Process management
- Process creation to deletion
- Processing resource requests
- Scheduling
- IPC
- Memory management
- I/O management
- Device management

PART-B

1. (i) Describe the three ways in which an RTOS handles the ISRs in a multitasking environment. (12)

1. **Direct call to ISR by an interrupting source (2)**

   - A hardware source calls an ISR directly.
   - ISR sends a message to the RTOS.
   - On an interrupt process running at the CPU is interrupted.
   - The ISR corresponding to that source starts execution.
   - Two Processes- ISR and RTOS.

   Diagram 1 (2)

2. **Direct call to RTOS by an interrupting source and temporary suspension of a scheduled task. (2)**

   - On interrupt the RTOS intercepts the hardware source call.
   - Initiates the corresponding ISR.
   - Sends one or more outputs and messages into queues and it must be short.
   - It is the task that runs the remaining code whenever it is scheduled.
   - RTOS schedules only the tasks.
   - ISR executes only during a temporary suspension of the task.

   Diagram 2 (2)

3. **Direct call to RTOS by an interrupting source and scheduling of tasks as well as ISRs by the RTOS. (2)**

   - The RTOS intercepts and executes the task needed on return from the ISR without any message for initiating the task from the ISR.
   - There are three i-th ISR, RTOS and j-th task in three memory blocks.
   - ISR_i executes after RTOS switches context.
   - The routine does not signed any message to the RTOS for initiating the i-th 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 called.
   - RTOS schedules not only the tasks but also the ISRs and switches.

   Diagram 3 (2)

ii) Enumerate the scheduling models used by RTOS schedulers. (4)

- Cyclic cooperative scheduling of ready tasks in a circular queue.

- Cooperative scheduling with precedence constraints.
- Cyclic cooperative scheduling with time slicing.
- Preemptive Scheduling.
- Fixed times Scheduling.
- Scheduling of periodic, sporadic and aperiodic task.
- Dynamic real time scheduling using Earliest Deadline First(EDF) precedence.
- Advanced Scheduling algorithms using probabilistic timed Petri nets or multi thread graphs

2. (i). Describe the cooperative round robin Scheduling using a circular queue of ready tasks. (8)

Cooperative: (1)

- Each ready task cooperates to let a running one finish.
- None of the tasks does a block any where during the ready to finish states.

Round robin: (1)

- Each ready task runs in turn only from the circular queue.
- The services in order in which a task is initiated on interrupt.

Cooperative round robin scheduling: (1)

- Each task has the same priority for execution in the round robin mode.
- The task priority parameter sets as per its position in the queue.
- Worst-case latency is same for each task.
- If a task is running all other ready task must wait.

A scheduler in which RTOS inserts into a list the ready task for a sequential execution in a cooperative round robin mode. (2)

Diagram 1

Program counter assignments at different time when the scheduler calls the tasks one by one in the circular list. (2)

Diagram 2

Worst-case Latency: (1)

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

Where,

$t_{ISR\text{-}>\ Sum}$ of all execution times for all the ISRs

T worst should always be less than the deadline.

(ii) Describe cyclic scheduling with time slicing (or) Describe rate monotonic cooperative scheduling? (8)

- When P^{th} task high execution time (et_p) worst case latency of the lowest priority task can exceed its deadline.
- RTOS defines a time slice for each task to finish and if it does not finish, then blocks its & finishes its remaining codes during the succeeding cycle.
- There is no insertion or deletion into the queue or limit. **(2)**

Diagram for programming model for cooperative time sliced scheduling. (2)

- Diagram for program counter on context switches between the schedule cells to tasks at two consecutive time slices.

$$T_{slice} >= T_{worst} \quad \textbf{(2)}$$

$$T_{worst} = \{ (dt_i + st_i + et_i)_1 + (dt_i + st_i + et_i)_2 + \ldots\ldots\ldots + (dt_i + st_i + et_i)_{N-1} + (dt_i + st_i + et_i)_N \} + t_{isr}$$

- If t_{slice} equals the sum of the max. time for each task then **(2)**
- Each task is executed once & finishes in one cycle itself.
- Waiting period
- Response Time.

3. Describe the preemptive scheduling model strategy. (16)

Disadvantages of cooperative scheduling: (1)

- Long execution time of a low priority task
- Cyclic but without a predefined $T_{Slice.}$

Preemptive scheduler: (2)

- Block a running task at the end of an instruction.
- The higher priority takes the control of the CPU.

Preemptive Scheduling of N Tasks:

Let the priority of task_1> task_2>task_3>task_4….>task N.

Diagram 1 (2)

Program counter assignments on the scheduler call to preempt task_2 when the priority of task_1> task_2>task_3

Diagram 1 (2)

- Each task has an infinite loop from start up to finish.
- Task_1 last instructions points to the pointed address, *next.
- Where it points to the return address to the RTOS.
- RTOS initiates and run the next task in the ready list.
- There is an RTOS message during the running of task_2 to preempt the task.

- The higher priority task_1 is initiated as follows. Task_2 blocks and sends a message to RTOS.
- The RTOS sends a message to task_1 to go to a ready state and run.
- A message is send to RTOS after task_1 finishes and the task_1 context becomes the same as at its start. An RTOS readies the task_2 and runs.
- A message will be sent to RTOS after task_2 finishes and task_2 context becomes same as at its start.
- RTOS message readies the task_3 and it will run. **(4)**

Advantage of using time out intervals: (2)

- Worst-case latency estimation is possible.
- The error reporting and handling by the RTOS.
- Provide a way to let the RTOS run even lowest priority task in necessary cases.

Conditions for preemption event (3)

- The preemption event takes place when an interrupt occurs and just before the return from the interrupt, there is a service call to the RTOS by ISR which set a token, the preemption event.
- Each RTOS uses a system clock ticking a RTCSWT. The preemption event takes place when an RTCSWT interrupt occurs at the RTOS. This event makes another higher priority task ready to run on the switch of the flag to the latter.
- The preemption event takes place when any call to the RTOS occur to enter the critical section or for sending the task message to the RTOS and if another higher priority task then needs to be serviced.

4. Explain in detail about critical section service by a preemptive scheduler. (16)

- A preemptive scheduler on arising out of the service need of the higher priority task should not preempt a lower priority task in certain cases.
- The blocking should not occur in the critical section of any lower priority task.

(1)

The Petri net for the task with a preemptive scheduling and one critical section where it takes a semaphore and release on critical section over:

Diagram 1 **(5)**

Places and Transitions: (10)

- The RTOS initiates idle to ready transition. RTOS sends two tokens: RTOS_create Event and taskjswitchflag.

- Consider the task_j_idleplace which currently has highest priority among the ready tasks.After the RTOS creates task_j, the place task_j_idle undergoes a transition to the ready state, task_j_ready place.

- When after taskj finisfes and when it is no longer needed under RTOS control, the RTOS sends a RTOS_DELETE event the task,it returns to the task_j_idle place and its corresponding task_j switch flag resets.
- At task_j_ready place the scheduler takes the priority parameter into account. If it is higher priority, it sets two tokens: taskjswitchflag=true and higher priority event=false for the transitions to the running task j place, task_j_running.

- From the task_j_running place the transition to the task_j_ready place will be fired when the task finish flag sets.
- At task_j_running place, the codes of the switched task j are executed.
- At the running task place, the transition for preempting will be fired when RTOS sends a token suspend event. Another enabling token if present, is time_out_event will also fired the transition.
- On a resume event the transition to task_j_running place occurs.
- At the task_j_running place there is another transition that fires so that the task_j is back at the to task_j_running place when the RTOS sends the token take_semaphore_event to ask the task j to take the semaphore**.**
- There can be none or one or several critical sections during the execution of a critical section. The RTOS resets the semaphore release flag and sets the take semaphore event token.

5. (i) Briefly explain about Fixed/Static Time Scheduling. (8)

Fixed Time Scheduling: (3)

- Fixed Time Scheduling is a scheduling strategy in which the time for each task is fixed.
- Let there be m tasks and m RTCSWTs, the scheduler can thus assign each task a fixed schedule.
- Each task undergoes a ready place to running place transition on the timeout of the corresponding timer.
- A scheduler is said to be using a fixed time scheduling method when the schedule is static and deterministic.
- The worst-case latencies for all the interrupts and tasks are pre-determinable.
- Coding for the tasks is such that execution times do not vary under the different inputs or different conditions.

Advantage: (1)

- No deadlines miss.

Methods to define Fixed Time Scheduling: (3)

Simulated annealing method.

- Heuristic method.
- Dynamic Programming model.

Example: (1)

- Situation in which a message for task is expected in a network from another system and the minimum and maximum for receiving it are unknown

(ii) Explain the functions and security features of kernel in OS. (8)

- Memory allocation
- Scheduling, Running and blocking of task.
- Handling of the hardware source call from the interrupt.
- Multitasking.
- Inter task Communication, synchronization.
- I/O management.
- Effective management of the multiple states of the CPU.
- Uses of semaphore by tasks.

UNIT – V

Real Time Operating System - Part II

Study of micro C/OS II or VX works or any other popular RTOS – RTOS system level functions – Task service functions – Time delay functions – Memory allocation related functions – Semaphore related functions – Mailbox related functions – Queue related functions – Case studies of programming with RTOS – Understanding case definition – Multiple tasks and their functions – Creating a list of tasks – Functions and IPCS – Exemplary coding steps.

PART - A (2 Marks Questions & Answers)

1.Define Vxworks Task.

Applications are organized into independent, though cooperating, programs. Each of these programs, while executing, is called a *task*. In VxWorks, tasks have immediate, shared access to most system resources, while also maintaining enough separate context to maintain individual threads of control.

2.How the task is Scheduled in Wind.

The default algorithm in *wind* is priority-based preemptive scheduling. The *wind* kernel has 256 priority levels, numbered 0 through 255. Priority 0 is the highest and priority 255 is the lowest. Tasks are assigned a priority when created. You can also change a task's priority level while it is executing by calling **taskPrioritySet( )**. The ability to change task priorities dynamically allows applications to track precedence changes in the real world.

3. List out the Functions of µC/OS-II.

- System level functions
- Task service functions
- Time delay functions
- Memory allocation related functions
- Semaphore related functions
- Mailbox related functions
- Queue related functions

4.What are the restrictions to be followed in hooked routines.

- Task switch hook routines must not assume any VM context is current other than the kernel context (as with ISRs).

- Task switch and swap hooks must not rely on knowledge of the current task or invoke any function that relies on this information
- A switch or swap hook must not rely on the **taskIdVerify(pOldTcb)** mechanism.

5.What is the alternate interface for message Queues?

Pipes provide an alternative interface to the message queue facility.Pipes are virtual I/O devices managed by the driver **pipeDrv**. The routine **pipeDevCreate( )** creates a pipe device and the underlying message queue associated with that pipe.

6.When will the Synchronous context switches occur.

Synchronous context switches occurs when executing task

- Makes a higher priority task ready to run
- Pends,delays,suspends itself
- Lower its own priority or exits

7.Give some WatchDog Calls.

- wdCreate( )

- wdStart( )
- wdDelete( )
- wdCancel( )

8.Give the mechanisms for Inter Process communication

- *Shared memory*, for simple sharing of data.
- *Semaphores*, for basic mutual exclusion and synchronization.
- *Mutexes* and *condition variables* for mutual exclusion and synchronization using POSIX interfaces.
- *Message queues* and *pipes*, for intertask message passing within a CPU.
- *Sockets* and *remote procedure calls*, for network-transparent intertask communication.
- *Signals*, for exception handling.

9. **Compare taskLock( ) and intLock( ).**

taskLock( )	**intLock( )**
No mutual exclusion is achieved	Mutual exclusion is achieved
If interrupted by hardware, the system will eventually return to task. If task ia blocked, it lose task lockout	If the call is too long, it can directly impact interrupt latency and cause the system to become far less deterministic.

10. How the task can be created in VxWorks?

The **taskSpawn( )** routine creates the new task context, which includes allocating the stack and setting up the task environment to call the main routine with the specified arguments. The new task begins execution at the entry to the specified routine.

11.Name some of the inter process communication function.

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

- semDelete()

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

- msgQCreate()
- msgQDelete()
- msgQSend()
- msgQReceive()

13.What are Vx Works pipes?

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

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

- Preemptive priority
- Time slicing

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

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

16. 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)

17. How is Vx Works TCB helpful for tasks?

- Provide 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.

18. 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.

19. 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.

20. 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.

PART - B (16 Marks Questions & Answers)

1. Explain RTOS System level functions in MUCOS.

- Used to initiation and start RTC ticks (interrupts) initiation and the ISR enter and exit functions
- For the critical section, MUCOS has interrupts disabling and enabling functions that execute at entering and exciting the section
- These are for task creating, running, suspending and resuming

Proto-type functions	**When is this OS function called**
Void OSInit(void)	Prior to OSStart()
Void OSStart(void)	After OSInit() & task creatint functions
Void OSTickInit (void)	In first task function that executes once to initialise the system timer ticks
Void OSInitEnter(void)	This is called just after the start of the ISR codes
Void OSInitExit(void)	This is called just before the return from the ISR
OS-ENTER-CRITICAL	To disable interrupts

OS-EXIT-CRITICAL	To enable interrupts

2 a). Explain Time delay functions in MUCOS.

Proto-type functions	**When is this OS function called**
OSTimeDly()	When a task is to be delayed by count inputs equal to the delay Count -1
OSTimeDlyResume()	When a task of priority equal to taskPriority is to resume before a preset delay
OSTimeDlyHMSM()	When need is to delay and blocks a task for hr hours, mn minutes, sec seconds and ms milliseconds

2 b). Explain Memory allocation related functions in MUCOS.

Proto-type functions	**When is this OS function called**
OSMemCreate()	To create an initialize a memory partition
OSMemGet()	to find pointers of the memory control block allocated to the memory blocks
OSMemPut()	To return a pointer of memory block in the memory partition from the memory control block pointer

OSMemQuery()	to find pointers of the memory control block OS_MemData data-structure.

3. Explain Mailbox related functions in MUCOS.

Proto-type functions	**When is this OS function called**
OS_Event OSMboxCreate()	To create an initialize a mailbox message pointer
OSMboxAccept()	To check whether mailbox message is available or not.
OSMboxPend()	To check whether mailbox message is pending or not.
OSMboxPost()	To check whether mailbox message is post or not.
OSMboxQuery()	To get mailbox error information

4. Explain Semaphore related functions in MUCOS.

Proto-type functions	**When is this OS function called**
OS_Event OSSemCreate()	To create an initialize a semaphore
OSSemPend()	To check whether semaphore is pending or not.

OSSemAccept()	To check whether semaphore is accept or not.
OSSemPost()	To check whether semaphore is post or not.
OSSemQuery()	To get semaphore information

5. Explain Queue related functions in MUCOS.

Proto-type functions	**When is this OS function called**
OS_Event OSQCreate()	To flush the previous queue and start with fresh queue input.
OSQFlush()	To check whether mailbox message is available or not.
OSQPend()	To elliminate all message in the queue that have been send. This function checke if the queue has a message pending at QMsgPointer.
OSQPost()	Sends a pointer of the message QMsg to the QMsgPointer at the queue back .
OSQPostFront()	Sends QMsg pointer to the QMsgPointer at the queue.
OSQQuery()	To get queue message's information and error information.

6. Explain VxWorks Facilities in detail.

- **High-Performance Real-time Kernel Facilities**

- The VxWorks kernel, *wind*, includes multitasking with preemptive priority scheduling, intertask synchronization and communications facilities, interrupt handling support, watchdog timers, and memory management.

❖ **POSIX Compatibility**

- VxWorks provides most interfaces specified by the 1003.1b standard (formerly the 1003.4 standard), simplifying your ports from other conforming systems.

❖ **Local File Systems**

- VxWorks provides fast file systems tailored to real-time applications. One file system is compatible with the MS-DOS® file system, another with the RT-11 file system, a third is a "raw disk" file system, a fourth supports SCSI tape devices, and a fifth supports CD-ROM devices.

❖ **C++ Development Support**

- In addition to general C++ support including the iostream library and the standard template library, the optional component Wind Foundation Classes adds the following C++ object libraries:
  - VxWorks Wrapper Class library
  - Tools.h++ library from Rogue Wave

❖ **Shared-Memory Objects (VxMP Option)**

- The VxMP option provides facilities for sharing semaphores, message queues, and memory regions between tasks on different processors.

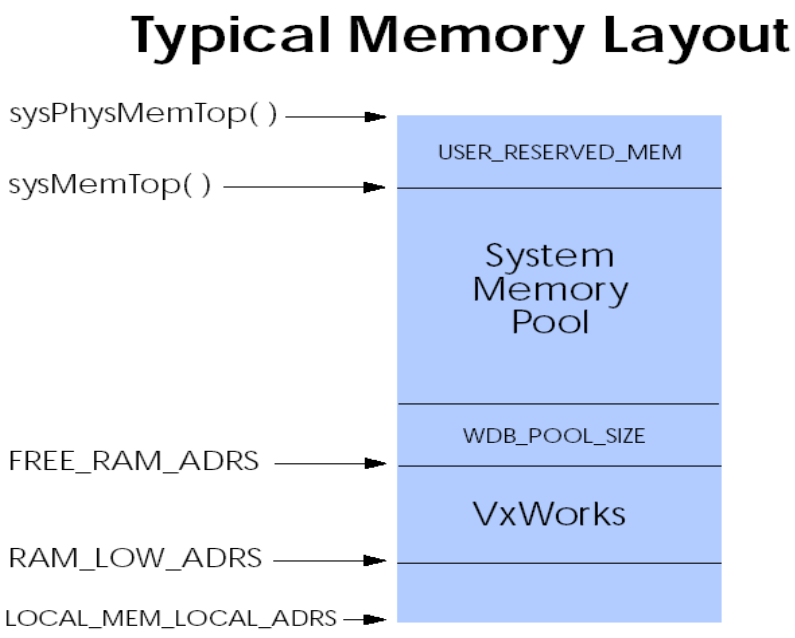
❖ **Virtual Memory (Including VxVMI Option)**

- VxWorks provides both bundled and unbundled (VxVMI) virtual memory support for boards with an MMU, including the ability to make portions of memory noncacheable or read-only, as well as a set of routines for virtual-memory management.
- Target-resident Tools
- In the Tornado development system, the development tools reside on the host system; see the *Tornado User's Guide* for details. However, a target-resident

shell, module loader and unloader, and symbol table can be configured into the VxWorks system if necessary.

- **Utility Libraries**
  - VxWorks provides an extensive set of utility routines, including interrupt handling, watchdog timers, message logging, memory allocation, string formatting and scanning, linear and ring buffer manipulations, linked-list manipulations, and ANSI C libraries.

7. Explain Memory Management in VxWorks .



System Memory Pool

- Used for dynamic memory allocation in programs:
  - *malloc( )*.
  - Creating tasks (stack and TCB).
  - VxWorks memory requests.
- Initialized at system start-up.
  - Can modify USER_RESERVED_MEM to reserve memory for application-specific use.
  - May need to modify *sysPhysMemTop( )* (or just LOCAL_MEM_SIZE) when adding memory to your board. Check your BSP documentation.

Target Server Memory Pool

- A pool of memory on the target reserved for use by the Tornado tools:
  - Dynamic loading of object modules.
  - Passing string arguments to tasks spawned on target.
  - Creation of variables from WindSh.
- The target server manages the pool, keeping overhead such as block lists on the host.
- The initial size of the target server memory pool is configured by WDB_POOL_SIZE. The default is 1/16 of **sysMemTop ( )** - FREE_RAM_ADRS.
- Additional memory is silently allocated from the system memory pool if needed.

Allocating/Releasing Memory

- To dynamically allocate memory:

8. Explain Semaphores in VxWorks.

VxWorks semaphores are highly optimized and provide the fastest intertask communication mechanism in VxWorks. Semaphores are the primary means for addressing the requirements of both mutual exclusion and task synchronization.

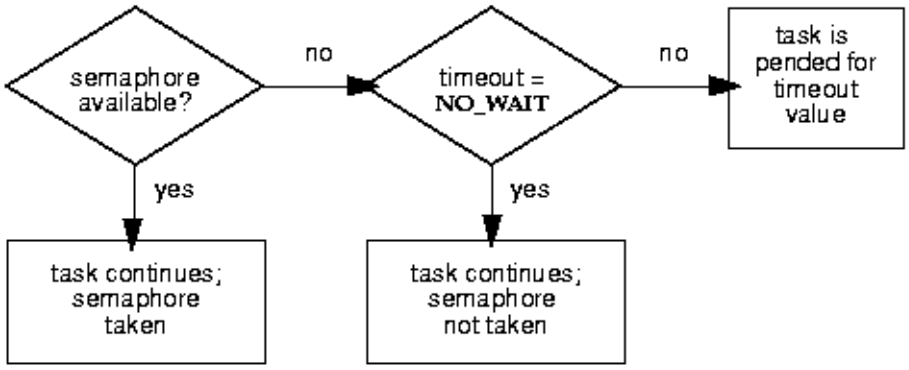
There are three types of semaphores

- *binary*
- *mutual exclusion*
- *counting*

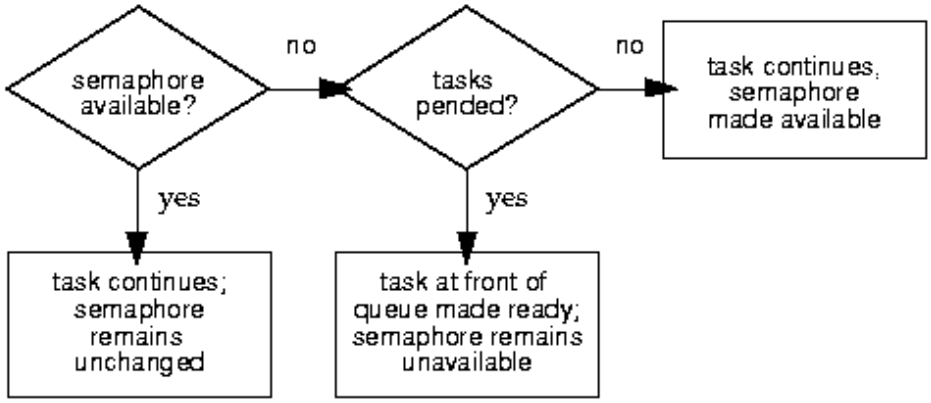
Binary Semaphores

A binary semaphore can be viewed as a flag. When a task takes a binary semaphore, the outcome depends on whether the semaphore is available (full) or unavailable (empty) at the time of the call.

Taking a Semaphore



Giving a Semaphore



Mutual-Exclusion Semaphores

The mutual-exclusion semaphore is a specialized binary semaphore designed to address issues inherent in mutual exclusion, including priority inversion, deletion safety, and recursive access to resources.

The fundamental behavior of the mutual-exclusion semaphore is identical to the binary semaphore, with the following exceptions:

- It can be used only for mutual exclusion.
- It can be given only by the task that took it.
- It cannot be given from an ISR.
- The **semFlush( )** operation is illegal.

Counting Semaphores

Counting semaphores are another means to implement task synchronization and mutual exclusion. Every time a semaphore is given, the count is incremented; every time a semaphore is taken, the count is decremented. When the count reaches zero, a task that tries to take the semaphore is blocked.

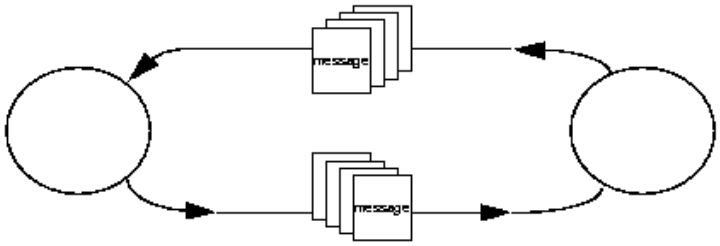
Semaphores and VxWorks Events

- semEvStart( )
- semGive( )
- semEvStop( )
- eventLib
- semEvStart( ).
- eventReceive( )

9. Explain Message Queues In VxWorks.

The primary intertask communication mechanism within a single CPU is *message queues*. Message queues allow a variable number of messages, each of variable length, to be queued. Tasks and ISRs can send messages to a message queue, and tasks can receive messages from a message queue.

Full Duplex Communication Using Message Queues



Two message-queue subroutine libraries:

- **msgQLib**,
- **mqPxLib**

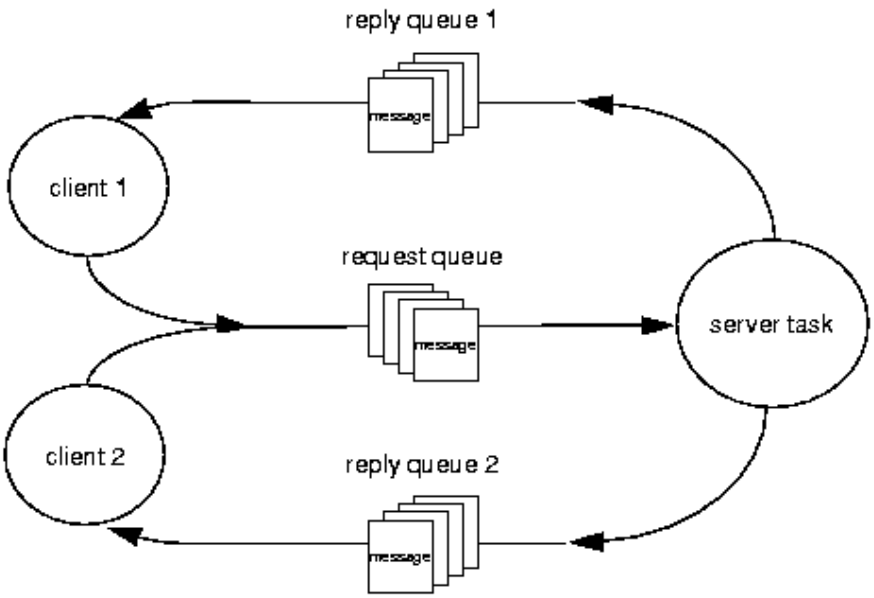
Wind Message Queues

Call	**Description**
msgQCreate( )	Allocates and initializes a message queue.
msgQDelete( )	Terminates and frees a message queue.
msgQSend( )	Sends a message to a message queue.
msgQReceive( )	Receives a message from a message queue.

Servers and Clients with Message Queues

In client-server model server tasks accept requests from client tasks to perform some service, and usually return a reply. The requests and replies are usually made in the form of intertask messages.

Client-Server Communications Using Message Queues



Message Queues and VxWorks Events

A message queue can send events to a task, if it is requested to do so by the task. To request that a message queue send events, a task must register with the message queue using **msgQEvStart( )**. To request that the message queue stop sending events, the registered task calls **msgQEvStop( )**.

Only one task can be registered with a message queue at any given time. The events a message queue sends to a task can be retrieved by the task using routines in **eventLib**. Details on when message queues send events are documented in the reference entry for **msgQEvStart( )**.

If the message queue is created with the **SG_Q_EVENTSEND_ERROR_NOTIFY** option, the send operation returns an error. Otherwise, VxWorks handles the error quietly.

Using **eventReceive( )**, a task may pend on events meant to be sent by a message queue. If the message queue is deleted, the task pending on events is returned to the ready state, just like the tasks that may be pending on the message queue itself.

10. Explain Case study of an Embedded System for a Smart Card.

Embedded Hardware :

Few basic features of the card hardware are as follows.

- Microcontroller used can be MC68HC11D0 or PIC16C84 or a smart card processor or an ASIP Processor
- Standard ROM is used.
- EEPROM of Flash is scalable
- RAM stores the temporary variables
- Chip Power Supply voltage extracts by a charge pump circuit
- IO System

Embedded Software :

Special features needed are as follows.

- Protected environment
- Restricted run time environment
- Its OS, every method, class and run time library should be scalable
- Code size generated should optimum
- Limited use of data types
- Three layered file system for the data
- Either a fixed length or variable file management
- Classes for the network, sockets, connections, data grams, character Input, output and streams, security management and so on.

List of task :

- Reset Task
- Task_Read Port
- Task_PW
- Task_Appl.