DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

QUESTION BANK

QUESTION BANK

UNIT-I

INTRODUCTION TO EMBEDDED SYSTEMS

Part - A (2 MARKS)

1. Define a System.
   A way of doing one or more tasks according to a program.

2. What is an embedded system?
   An embedded system employs a combination of hardware & software (a "computational engine") to perform a specific function; is part of a larger system that may not be a "computer"; works in a reactive and time-constrained environment

3. What are the typical characteristics of an embedded system?
   Typical characteristics:
   Perform a single or tightly knit set of functions;
   Increasingly high-performance & real-time constrained;
   Power, cost and reliability are often important attributes
   That influence design;
   Application specific processor design can be a significant component of some embedded systems.
   Other characteristics:
   - Application specific
   - Digital signal processing in ECS
   - Reactive
   - Real-time
   - Distributed

4. What are the advantages of embedded system?
   Advantages:
   Customization yields lower area, power, cost, etc.,

5. What are the disadvantages of embedded system?
   Disadvantages:
   Higher HW/software development overhead design, compilers, debuggers, etc., may result in delayed time to market!

6. What are the applications of an embedded system?

Embedded Systems: Applications:

- Consumer electronics, e.g., cameras, camcorders, etc.,
- Consumer products, e.g., washers, microwave ovens, etc.,
- Automobiles (anti-lock braking, engine control, etc.,)
- Industrial process controllers & avionics/defense applications
- Computer/Communication products, e.g., printers, FAX machines, etc.,
- Emerging multimedia applications & consumer electronics

7. What are the various embedded system designs Modeling Refining (or "partitioning") HW-SW partitioning

8. What are the complicating factors in embedded design?

Complicating factors in the design of embedded systems

- Many of the subtasks in design are intertwined.
- Allocation depends on the partitioning, and scheduling presumes a certain allocation.
- Predicting the time for implementing the modules in hardware or software is not very easy, particularly for tasks that have not been performed before.

9. What are the real-time requirements of an embedded system?

Hard-real time systems: where there is a high penalty for missing a deadline e.g., control systems for aircraft/space probes/nuclear reactors; refresh rates for video, or DRAM. Soft realtime systems: where there is a steadily increasing penalty if a deadline is missed.

e.g., laser printer: rated by pages-per-minute, but can take differing times to print a page (depending on the \"complexity\" of the page) without harming the machine or the customer.

10. What are the functional requirements of embedded system?

Data Collection

- Sensor requirements
- Signal conditioning
- Alarm monitoring

Direct Digital Control

- Actuators

Man-Machine Interaction

- informs the operator of the current state of the controlled object
- assists the operator in controlling the system.

11. What are the main components of an embedded system?

Three main components of embedded systems:

1. The Hardware
2. Application Software
3. RTOS

12. Define embedded microcontroller.

An embedded microcontroller is particularly suited for embedded applications to perform dedicated task or operation.

Example: 68HC11xx, 8051, PIC, 16F877, etc.,

13. Explain digital signal processing in embedded system continued digitization of signals increasing the role of DSP in ES.
    - Signals are represented digitally as sequence of "samples"
    - ADC's are moving closer to signals

14. What are the various classifications of embedded systems?
    1. Small scale embedded systems
    2. Medium scale embedded systems
    3. Sophisticated embedded systems

15. What are the two essential units of a processor on an embedded system?
    1. Program flow control unit (CU)
    2. Execution unit (EU)

16. What does the execution unit of a processor in an embedded system do?
    The execution unit implements data transfer and data conversion. It includes ALU and circuits that execute instruction for jump, interrupt, etc.,

17. Give examples for general purpose processor.
1. Microprocessor
2. Microcontroller
3. Embedded processor
4. Digital Signal Processor
5. Media Processor

18. Define microprocessor.
    A microprocessor fetches and processes the set of general-purpose instructions such as data transfer, ALU operations, stack operations, I/O operations and other program control operations.

19. When is Application Specific System processors (ASSPs) used in an embedded system?
    An ASSP is dedicated to real-time video processing applications such as video conferencing, video compression and decompression systems. It is used as an additional processing unit for running application specific tasks in the place of processing using embeddedsoftware.

20. What is the need for LCD and LED displays?
    Uses of LCD and LED display:
    1. It is used for displaying and messaging.
    2. Example: Traffic light status indicator, remote controls,signals, etc.,
    3. The system must provide necessary circuit and software for the output to LCD controller.

21. Define ROM image.
    ROM image in a system memory consists of:
    Boot-up program, stack address pointer, program counter address pointer, application tasks, ISRs, input data, RTOS and vector addresses.
    Bytes at each address must be defined to create ROM image.

22. Define device driver.
A device driver is software for controlling, reading, sending a byte of stream of bytes from/to the device.

23. Give some examples for small scale embedded systems.
68HC05, PIC 16F8x, 8051, etc.,

24. Give some examples for medium scale embedded systems
8051, 80251, 80x86, 80196,68HC11xx

25. Give some examples for sophisticated embedded systems
ARM7, Power PC, Intel 80960, etc.,

Other Important Questions

26. Give the reactivity in embedded system.
Closed systems
- Execution indeterminacy confined to one source
- Causal relations are easily established.

Open systems
- Indeterminacy from multiple sources, not controllable or observable by the programmer not possible to infer causal relations

27. Explain distributed systems.
- Consist of components that may necessarily be physically distributed.
- Consist of communicating processes on multiple processors and/or dedicated hardware connected by Communication links.
- Motivation:
  economical
  multiple processors to handle multiple time-critical tasks
  physically distributed
  Devices under control may be physically distributed.

28. What are the various embedded system requirements
Types of requirements imposed by embedded applications:
- R1 Functional requirements
- R2 Temporal requirements
- R3 Dependability requirements

29. What are the temporal requirements?
Tasks may have deadlines
- Minimal latency jitter
- Minimal error detection latency
- Timing requirements due to tight software control loops
- Human interface requirements.

30. Give the classification of embedded system.
- Multi-dimensional classifications
- Hard versus software systems

- Fail-safe versus fail-operational systems
- Guaranteed-response versus best-effort
- Resource-adequate versus resource-inadequate
- Event-triggered versus time-triggered.

31. What are the types of programmable logic?
    PLDs,CPLDs,FPGAs

32. Explain the VLSI design Flow.
    The design process, at various levels, is usually evolutionary in nature. It starts with a given set of requirements. Initial design is developed and tested against the requirements. When requirements are not met, the design has to be improved. If such improvement is either not possible or too costly, then the revision of requirements and its impact analysis must be considered. The Y-chart (first introduced by D.Gajski) shown in Fig. 1.4 illustrates a design flow for most logic chips, using design activities on three different axes (domains) which resemble the letter Y.

33. Give the VLSI design hierarchy
    The use of hierarchy or "divide and conquer" technique involves dividing a module into sub- modules and then repeating this operation on the sub-modules until the complexity of the smaller parts becomes manageable. This approach is very similar to the software case where large programs are split into smaller and smaller sections until simple subroutines,

34. What are embedded cores?
    More and more vendors are selling or giving away their processors and peripherals in a form that is ready to be integrated into a programmable logic-based design. They both recognize the potential for growth in the system-on-a-chip area and want a piece of the royalties or want to promote the use of their particular FPGA or CPLD by providing libraries of ready-to-use building blocks. Either way, you will gain with lower system costs and faster time-to-market.

35. What are hybrid chips?
    The vendors of hybrid chips are betting that a processor core embedded within a programmable logic device will require far too many gates for typical applications. So they\'ve created hybrid chips that are part fixed logic and part programmable logic. The fixed logic contains a fully functional processor and perhaps even some on-chip memory. This part of the chip also interfaces to dedicated address and data bus pins on the outside of the chip. Application-specific peripherals can be inserted into the programmable logic portion of the chip, either from a library of IP cores or the customer\'s own designs.

36. What do you meant by gatecounts?
    The gate count by itself is almost useless. Different vendors use different measures: number of available gates, equivalent number of NAND gates, equivalent number of gates in a PLD, equivalent number of gates in an ASIC, etc. You simply can\'t compare these numbers across vendors. A better comparison can be made in terms of numbers of registers (flip-flops) and I/O pins.

37. What is prototyping?
    Many times a CPLD or FPGA will be used in a prototype system. A small device may be present to allow the designers to change a board\'s glue logic more easily during product

development and testing. Or a large device may be included to allow prototyping of a system-on-a-chip design that will eventually find its way into an ASIC. Either way, the basic idea is the same: allow the hardware to be flexible during product development. When the product is ready to ship in large quantities, the programmable device will be replaced with a less expensive, though functionally equivalent, hard-wired alternative

38. Give the internal structure of FPGA

The development of the FPGA was distinct from the PLD/CPLD evolution just described. This is apparent when you look at the structures inside. Figure 2 illustrates a typical FPGA architecture. There are three key parts of its structure: logic blocks, interconnect, and I/O blocks. The I/O blocks form a ring around the outer edge of the part. Each of these provides individually selectable input, output, or bi-directional access to one of the general-purpose I/O pins on the exterior of the FPGA package. Inside the ring of I/O blocks lies a rectangular array of logic blocks. And connecting logic blocks to logic blocks and I/O blocks to logic blocks is the programmable interconnect wiring.

39. Define FPGAs

Field Programmable Gate Arrays (FPGAs) can be used to implement just about any hardware design. One common use is to prototype a lump of hardware that will eventually find its way into an ASIC.

40. Define PLDs

At the low end of the spectrum are the original Programmable Logic Devices (PLDs). These were the first chips that could be used to implement a flexible digital logic design in hardware. In other words, you could remove a couple of the 7400-series TTL parts (ANDs, ORs, and NOTs) from your board and replace them with a single PLD. Other names you might encounter for this class of device are Programmable Logic Array (PLA), Programmable Array Logic (PAL), and Generic Array Logic (GAL).

41. What are dependability requirements of an embedded system?

- Safety
- Critical failure modes
- Certification
- Maintainability
- MTTR in terms of repairs per hour
- Availability
- A = MTTF / (MTTF + MTTR)
- Security

42. Give the diversity of embedded computing

Diversity in Embedded Computing
Pocket remote control RF transmitter 100 KIPS, crush-proof, long battery life
Software optimized for size
Industrial equipment controller
1 MIPS, safety-critical, 1 MB memory
Software control loops
Military signal processing
1 GFLOPS, 1 GB/sec IO, 32 MB

Part – B (16 Marks)

1. What is the need for IDE in an Embedded Architecture? Discuss.
2. Explain the various forms of memory and the functions assigned to them.
3. Explain the software embedded systems
4. Explain the components of exemplary embedded systems
5. Describe the architecture of a typical micro controller with a neat diagram.
6. Explain the basic processors and hardware units in the embedded system
7. Explain how software is embedded into a system
8. Explain the methods used in the embedded system on a chip
9. List the hard ware units that must be present in the embedded systems.
10. i) Explain the Exemplary applications of each type of embedded system.
    ii) Explain the different program layers in the embedded software and also the process of converting a "C" program into the file for ROM image with suitable block diagrams.
11. Explain the Embedded System on Chip (SoC) & in VLSI circuit.
12. i) Explain the various form of memories present in a system
    ii) Explain the software tools in designing of an embedded system.

Subject Code: EC1308 Year/ Sem:IV /VII
Subject Name: Embedded Systems Branch : ECE

UNIT-II
DEVICES AND BUSES FOR DEVICES NETWORK
Part - A (2 MARKS)

1. Give the summary of I/O devices used in embedded system

Program, data and stack memories occupy the same memory space. The total addressable memory size is 64 KB.

Program memory - program can be located anywhere in memory. Jump, branch and call instructions use 16-bit addresses, i.e. they can be used to jump/branch anywhere within 64 KB. All jump/branch instructions use absolute addressing.

Data memory - the processor always uses 16-bit addresses so that data can be placed anywhere. Stack memory is limited only by the size of memory. Stack grows downward. First 64 bytes in a zero memory page should be reserved for vectors used by RST instructions.

I/O ports

256 Input ports

256 Output ports

Registers

Accumulator or A register is an 8-bit register used for arithmetic, logic, I/O and load/store operations.

2. Define bus.

Buses: The exchange of information.

Information is transferred between units of the microcomputer by collections of conductors called buses.

There will be one conductor for each bit of information to be passed, e.g., 16 lines for a 16 bit address bus. There will be address, control, and data buses

3. What are the classifications of I/O devices?

i. Synchronous serial input and output
ii. Asynchronous serial UART input and output
iii. Parallel one bit input and output
iv. Parallel port input and output

4. Give some examples for serial input I/O devices.
Audio input, video input, dial tone, transceiver input, scanner, serial IO bus input, etc.,

5. Give the steps for accomplishing input output data transfer
Accomplishing input/output data transfer
There are three main methods used to perform/control input/output data transfers. They are,
- Software programming (scanning or polling)
- interrupt controlled
- Direct memory access (DMA)

6. Give the limitations of polling technique.
The polling technique, however, has limitations.
- It is wasteful of the processors time, as it needlessly checks the status of all devices all the time.
- It is inherently slow, as it checks the status of all I/O devices before it comes back to check any given one again.
- When fast devices are connected to a system, polling may simply not be fast enough to satisfy the minimum service requirements. Priority of the device is determined

7. What do you meant by bus arbitration?
Bus Arbitration
Most processors use special control lines for bus arbitration, ie, controlling the use of the address and data bus,
- An input which the DMAC uses to request the bus
- An output(s) indicating the bus status
- An output indicating acceptance of the DMAC\'s bus request

8. What are the two characteristics of synchronous communication?
- Bytes/frames maintain constant phase difference and should not be sent at random time intervals. No handshaking signals are provided during the communication.
- Clock pulse is required to transmit a byte or frame serially. Clock rate information is transmitted by the transmitter.

9. What do you mean by asynchronous communication?
The most basic way of sharing data is by copying the data in question to each server. This will only work if the data is changed infrequently and always by someone with administrative access to all the servers in the cluster.

10. What are the characteristics of asynchronous communication?
- Variable bit rate - need not maintain constant phase difference
- Handshaking method is used
- Transmitter need not transmit clock information along with data bit stream

11. What are the three ways of communication for a device?
i. Separate clock pulse along with data bits
ii. Data bits modulated with clock information
iii. Embedded clock information with data bits before transmitting

12. Expand a) SPI b) SCI
SPI - SERIAL PERIPHERAL INTERFACE

SCI - SERIAL COMMUNICATION INTERFACE

13. What are the features of SPI?
    - SPI has programmable clock rates
    - Full-duplex mode
    - Crystal clock frequency is 8MHz
    - Open drain or totempole output from master to slave

14. Define software timer.
    A software timer is software that executes the increase/decrease count value on an interrupt from timer or RTC. Software timer is used as virtual timing device.

15. What are the forms of timer?
    - Hardware interrupt timer
    - Software timer
    - User software controlled hardware timer
    - RTOS controlled hardware timer
    - UP/DOWN count action timer
    - One-shot timer (No reload after overflow and finished states)

16. Define RTC
    RTC Stands for Real Time Systems. Once the system starts, do not stop/reset and the count value cannot be reloaded.

17. What is I2C?
    Inter- Integrated Circuit (2-wire/line protocol) which offers synchronous communication. Standard speed: 100Kbps and High speed: 400 Kbps

18. What are the bits in I2C corresponding to?
    SDA - Serial Data Line and SCL - Serial Clock line

19. What is a CAN bus? Where is it used?
    CAN stands for Controller Area Network. Serial line, bi-directional bus used in automobiles. Operates at the rate of 1Mbps.

20. What is USB? Where is it used?
    USB - Universal Serial Bus
    Operating speed - upto 12 Mbps in fast mode and 1.5Mbps in low-speed mode.

21. What are the features of the USB protocol?
    A device can be attached, configured and used, reset, reconfigured and used, detached and reattached, share the bandwidth with other devices.

22. What are the four types of data transfer used in USB?
    - Controlled transfer
    - Bulk transfer
    - Interrupt driven data transfer
    - Iso-synchronous transfer

23. Explain briefly about PCI and PCI/X buses.
    - Used for most PC based interfacing
    - Provides superior throughput than EISA
    - Platform-independent
    - Clock rate is nearest to sub-multiples of system clock

24. Mention some advanced bus standard protocols;
    1. GMII (Gigabit Ethernet MAC Interchange Interface)
    2. XGMI (10 Gigabit Ethernet MAC Interchange Interface)
    3. CSIX-1 6.6 Gbps
    4. Rapid IO interconnect specification v1.1 at 8 Gbps

25. What do you meant by high speed device interfaces?
    Fail-over clustering would not be practical without some way for the redundant servers to access remote storage devices without taking a large performance hit, as would occur if these devices were simply living on the local network. Two common solutions to this problem are double-ended SCSI and fibre-channel.

26. Mention some I/O standard interfaces.
    HSTL - High Speed Transceiver Logic (Used in high speed operations)
    SSTL - Stub Series Terminated Logic (Used when the buses are needed to isolate from the large no. of stubs)

Part – B (16 Marks)

1. Describe the functions of a typical parallel I/O interface with a neat diagram
2. Explain high speed I/O interfacing in detail
3. Write short notes on
    (i) Analog to digital converter
    (ii) UART
4. Explain the functions of various buses used during transfer
5. Explain the synchronous and asynchronous communications from serial devices
6. Explain the various timer and counting devices
7. Explain the various bus structures used in embedded systems
8. Explain the sophisticated interfacing features in devices /ports
9. Explain the classification of IO devices.
10. Explain the working of timers and counters in detail.
11. Explain the serial communication using I2C, CAN, USB in detail.
12. i) Explain the parallel port devices.
    ii) Explain the sophisticated interfacing features in device ports.
13. i) Explain the signal using a transfer of byte when using the I2C bus and also the format of bits at the I2C bus with diagram.
    ii)Explain the internal serial communication devices.
14. Explain the following parallel communication devices:
    i) ISA bus
    ii) PCI and PCI/X

QUESTION BANK

Subject Code: EC1308 Year/ Sem:IV /VII
Subject Name: Embedded Systems Branch : ECE

UNIT-III
PROGRAMMING CONCEPTS
Part - A (2 MARKS)

1. What are the advantages of Assembly language?
   - It gives the precise control of the processor internal devices and full use of processor specific features in its instruction sets and addressing modes.
   - The machine codes are compact, which requires only small memory.
   - Device drivers need only few assembly instructions.

2. What are advantages of high level languages?
   - Data type declaration
   - Type checking
   - Control structures
   - Probability of non-processor specific codes

3. Define In -line assembly
   Inserting an assembly code in between is said to be in-line assembly.

4. Mention the elements of C program.
   1. Files:
      1. Header files
      2. Source files
      3. Configuration files
      4. Preprocessor directives
   2. Functions:
      1. Macro function
      2. Main function
      3. Interrupt service routines or device drivers
   3. Others:
      1. Data types
      2. Data structures
      3. Modifiers
      4. Statements

5. Loops and pointers

5. What is the use of MACRO function?
   - A macro function executes a named small collection of codes, with the values passed by the calling function through its arguments.
   - It has constant saving and retrieving overheads.

6. What is the use of interrupt service routines or device drivers?
   - It is used for the declaration of functions and datatypes, typedef and executes named set of codes.
   - ISR must be small (short), reentrant or must have solution for shared data problem.

7. What are the datatypes available in C language?
   Char - 8 bit; byte - 8 bit; short - 16 bit; unsigned short - 16 bit; unsigned int - 32 bit; int - 32 bit; long double - 64 bit; float - 32 bit; double - 64

8. Mention the data structures available in C language.
   1. Queue
   2. Stack
   3. Array (1-dimentional and multi-dimentional)
   4. List
   5. Tree
   6. Binary-tree

9. Write the syntax for declaration of pointer and Null-pointer.
   Syntax for pointer:
   ```
   void *portAdata
   ```
   Syntax for Null-pointer:
   ```
   #define NULL (void*) 0x0000
   ```

10. Explain pass by values.
    - The values are copied into the arguments of the function.
    - Called programs does not change the values of the variables

11. What are the three conditions that must be satisfied by the re-entrant function?
    1. All the arguments pass the values and none of the argument is a pointer.
    2. When a non-atomic operation, that function should not operate on the function declared outside.
    3. A function does does not call a function by itself when it is not reentrant.

12. Explain pass by reference.
    - When an argument value to a function is passed through a pointer, then the value can be changed.
    - New value in the calling function will be returned from the called function.

13. Write the syntax for function pointer.
    Syntax:
    ```
    void *<function_name> (function arguments)
    ```

14. Define queue.
    - A structure with a series of elements.
    - Uses FIFO mode.
    - It is used when an element is not directly accessed using pointer and index but only through FIFO.
    - Two pointers are used for insertion and deletion.

15. Define stack.
    - A structure with a series of elements which uses LIFO mode.
    - An element can be pushed only at the top and only one pointer is used for POP.
    - Used when an element is not accessible through pointer and index, but only through LIFO.

16. Define List.
    - Each element has a pointer to its next element.
    - Only the first element is identifiable and it is done using list-top pointer (header).
    - Other element has no direct access and is accessed through the first element.

17. What is Object oriented programming?
    An object-oriented programming language is used when there is a need for re-usability of defined objects or a set of objects that are common for many applications.

18. What are the advantages of OOPs?
    - Data encapsulation
    - Reusable software components
    - inheritance

19. What are the characteristics of OOPs?
    - An identity - reference to a memory block
    - A state - data, field and attributes
    - A behavior - methods to manipulate the state of the object

20. Define Class.
    A class declaration defines a new type that links code and data. It is then used to declare objects of that class. Thus a class is an logical abstraction but an object has physical existence.

21. Define NULL function
    NULL defines empty stack or no content in the stack/queue/list.

22. What is Multiple Inheritance?
    Inheritance is the process by which objects of one class acquire the properties of objects of another class. In OOP, the concept of inheritance provides the idea of reusability.

23. Define Exception handling
    Exceptions are used to report error conditions. Exception handling is built upon three keywords:
    1. try

2. catch
3. throw

24. What is a Preprocessor Directive?

A preprocessor directive starts with '#' sign. The following are the types of preprocessor directives:

1. Preprocessor global variables
2. Preprocessor constants

25. Mention the flags available for queue.

1. QerrrorFlag
2. HeaderFlag
3. TrailingFlag
4. cirQuFlag
5. PolyQuFlag

Part – B (16 Marks)

1. i) Tabulate program elements: Macros and Functions and their uses. (8)
   ii) Explain the use of pointers, NULL pointers (8)
2. i) Explain the multiple function calls in the cyclic order in the main. Also write the advantages of building ISR queues. Explain (8)
   ii) Explain the 'C' program compiler and cross compiler. (8)
3. i) Explain the optimization of memory codes. (8)
   ii) Explain the Embedded programming in C++. (8)
4. Explain the function pointers, function queues and ISR queues. (16)

QUESTION BANK

Subject Code: EC1308 Year/ Sem:IV /VII
Subject Name: Embedded Systems Branch : ECE

UNIT-IV
REAL TIME OPERATING SYSTEMS - PART I
Part - A (2 MARKS)

1. Define process.
A process is a program that performs a specific function.

2. Define task and Task state.
A task is a program that is within a process. It has the following states:
1. Ready
2. Running
3. Blocked
4. Idle

3. Define (TCB)
The TCB stands for Task Control Block which holds the control of all the tasks within the block. It has separate stack and program counter for each task.

4. What is a thread?
A thread otherwise called a lightweight process (LWP) is a basic unit of CPU utilization, it comprises of a thread id, a program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and operating system resources such as open files and signals.

5. What are the benefits of multithreaded programming?
The benefits of multithreaded programming can be broken down into four major categories:
¬ Responsiveness
¬ Resource sharing
¬ Economy
¬ Utilization of multiprocessor architectures

6. Compare user threads and kernel threads.
User threads Kernel threads

User threads are supported above the kernel and are implemented by a thread library at the user level Kernel threads are supported directly by the operating system
Thread creation & scheduling are done in the user space, without kernel intervention. Therefore they are fast to create and manage Thread creation, scheduling and management are done by the operating system. Therefore they are slower to create & manage compared to user threads Blocking system call will cause the entire process to block If the thread performs a blocking system call, the kernel can schedule another thread in the application for execution

7. Define RTOS.

A real-time operating system (RTOS) is an operating system that has been developed for real-time applications. It is typically used for embedded applications, such as mobile telephones, industrial robots, or scientific research equipment.

8. Define task and task rates.

An RTOS facilitates the creation of real-time systems, but does not guarantee that they are real-time; this requires correct development of the system level software. Nor does an RTOS necessarily have high throughput — rather they allow, through specialized scheduling algorithms and deterministic behavior, the guarantee that system deadlines can be met. That is, an RTOS is valued more for how quickly it can respond to an event than for the total amount of work it can do. Key factors in evaluating an RTOS are therefore maximal interrupt and thread latency

9. Define CPU scheduling.

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multi-programmed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

10. Define Synchronization.

Message passing can be either blocking or non-blocking. Blocking is considered to be synchronous and non-blocking is considered to be asynchronous.

11. Define Inter process communication.

Inter-process communication (IPC) is a set of techniques for the exchange of data among multiple threads in one or more processes. Processes may be running on one or more computers connected by a network. IPC techniques are divided into methods for message passing, synchronization, shared memory, and remote procedure calls (RPC). The method of IPC used may vary based on the bandwidth and latency of communication between the threads, and the type of data being communicated.

12. Define Semaphore.

A semaphore ‘S’ is a synchronization tool which is an integer value that, apart from initialization, is accessed only through two standard atomic operations; wait and signal. Semaphores can be used to deal with the n-process critical section problem. It can be also used to solve various synchronization problems.

The classic definition of ‘wait’

```
wait (S){
while    (S<=0)
;
S--;
```

```
}
The classic definition of 'signal'
signal (S){
S++;
}
```

13. What is a semaphore?

Semaphores -- software, blocking, OS assistance solution to the mutual exclusion problem basically a non-negative integer variable that saves the number of wakeup signals sent so they are not lost if the process is not sleeping another interpretation we will see is that the semaphore value represents the number of resources available

15. Give the semaphore related functions.

A semaphore enforces mutual exclusion and controls access to the process critical sections. Only one process at a time can call the function fn.
SR Program: A Semaphore Prevents the Race Condition.
SR Program: A Semaphore Prevents Another Race Condition.

16. When the error will occur when we use the semaphore?

i. When the process interchanges the order in which the wait and signal operations on the semaphore mutex.
ii. When a process replaces a signal (mutex) with wait (mutex).
iii. When a process omits the wait (mutex), or the signal (mutex), or both.

17. Differentiate counting semaphore and binary semaphore.

Binary Semaphore:

The general-purpose binary semaphore is capable of addressing the requirements of both forms of task coordination: mutual exclusion and synchronization.

A binary semaphore can be viewed as a flag that is available (full) or unavailable (empty). Counting semaphores are another means to implement task synchronization and mutual exclusion.

Counting Semaphore:

The counting semaphore works like the binary semaphore except that it keeps track of the number of times a semaphore is given. 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. As with the binary semaphore, if a semaphore is given and a task is blocked, it becomes unblocked. However, unlike the binary semaphore, if a semaphore is given and no tasks are blocked, then the count is incremented. This means that a semaphore that is given twice can be taken twice without blocking.

18. What is priority inheritance?

Priority inheritance is a method for eliminating priority inversion problems. Using this programming method, a process scheduling algorithm will increase the priority of a process to the maximum priority of any process waiting for any resource on which the process has a resource lock.

19. Define Message Queue.

A message queue is a buffer managed by the operating system. 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 (if it is nonempty). Queues can use a FIFO (First In, First Out) policy or it can be based on priorities.

Message queues provide an asynchronous communications protocol.

20. Define Mailbox and Pipe.

A mailboxes are software-engineering components used for interprocess communication, or for inter-thread communication within the same process. A mailbox is a combination of a semaphore and a message queue (or pipe).

Message queue is same as pipe with the only difference that pipe is byte oriented while queue can be of any size.

21. Define Socket.

A socket is an endpoint for communications between tasks; data is sent from one socket to another.

22. Define Remote Procedure Call.

Remote Procedure Calls (RPC) is a facility that allows a process on one machine to call a procedure that is executed by another process on either the same machine or a remote machine. Internally, RPC uses sockets as the underlying communication mechanism.

Other Important Questions

23. . Define thread cancellation & target thread.

The thread cancellation is the task of terminating a thread before it has completed. A thread that is to be cancelled is often referred to as the target thread. For example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be cancelled.

27. What are the different ways in which a thread can be cancelled?

Cancellation of a target thread may occur in two different scenarios:

¬ Asynchronous cancellation: One thread immediately terminates the target thread is called asynchronous cancellation.

¬ Deferred cancellation: The target thread can periodically check if it should terminate, allowing the target thread an opportunity to terminate itself in an orderly fashion.

28. What is preemptive and non-preemptive scheduling?

• Under non-preemptive scheduling once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or switching to the waiting state.

• Preemptive scheduling can preempt a process which is utilizing the CPU in between its execution and give the CPU to another process.

29. What is a Dispatcher?

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves:

¬ Switching context

¬ Switching to user mode

¬ Jumping to the proper location in the user program to restart that program.

30. What is dispatch latency?
The time taken by the dispatcher to stop one process and start another running is known as dispatch latency.

31. What are the various scheduling criteria for CPU scheduling?
The various scheduling criteria are
- ¬ CPU utilization
- ¬ Throughput
- ¬ Turnaround time
- ¬ Waiting time
- ¬ Response time

32. Define throughput?
Throughput in CPU scheduling is the number of processes that are completed per unit time. For long processes, this rate may be one process per hour; for short transactions, throughput might be 10 processes per second.

33. What is turnaround time?
Turnaround time is the interval from the time of submission to the time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

34. Define race condition.
When several process access and manipulate same data concurrently, then the outcome of the execution depends on particular order in which the access takes place is called race condition. To avoid race condition, only one process at a time can manipulate the shared variable.

35. What is critical section problem?
Consider a system consists of 'n' processes. Each process has segment of code called a critical section, in which the process may be changing common variables, updating a table, writing a file. When one process is executing in its critical section, no other process can allowed executing in its critical section.

36. What are the requirements that a solution to the critical section problem must satisfy?
The three requirements are
- ¬ Mutual exclusion
- ¬ Progress
- ¬ Bounded waiting

37. Define deadlock.
A process requests resources; if the resources are not available at that time, the process enters a wait state. Waiting processes may never again change state, because the resources they have requested are held by other waiting processes. This situation is called a deadlock.

38. What are conditions under which a deadlock situation may arise?
A deadlock situation can arise if the following four conditions hold simultaneously in a system:
1. Mutual exclusion

2. Hold and wait
3. No pre-emption
4. Circular wait

39. What are the various shared data operating system services?
• explain how operating systems provide abstraction from the computer hardware.
• describe the meaning of processes, threads and scheduling in a multitasking operating system.
• describe the role of memory management explaining the terms memory swapping, memory paging, and virtual memory.
• contrast the way that MS-DOS and unix implement file systems compare the design of some real operating systems.

Part – B (16 Marks)

1. Explain how thread and process are used in embedded system.
2. Explain process management and memory management in embedded system
3. Explain file system organization and implementation
4. Explain how interrupt routines handled in embedded system
6. Explain the real time operating systems
7. Explain cyclic scheduling with time slicing
8. Explain how critical section in handled by a pre-emptive scheduler
9. Explain interprocess communication and synchronization
10. Explain interprocess communications using signals
11. Explain remote procedure call with an example
12. Write about IPC in detail.
13. a) Write about Interrupt Service Handling in RTOS.
b) Write about Semaphores with types in detail.
14. Explain state transition diagram of RTOS.
15. What are the rules to decide reentrancy of a function? Explain due to which rule the following function is not reentrant?

```
int cErrors;
void vcountErrors(int cNewError) {
cErrors += cNewError;
}
```

Can this function be made reentrant by using semaphore? Justify.

16. i) Explain the goals of operating system services.
ii) Explain the three alternative systems in three RTOS for responding a hardware source call with the diagram.
17. i) Explain the scheduler in which RTOS insert into the list and the ready task for sequential execution in a co-operative round robin model.
ii) Explain the fifteen point strategy for synchronization between the processes, ISRs, OS functions and tasks for resource management.
18. i) Explain the critical section service by a preemptive scheduler.
ii) Explain the Rate Monotonic Co-operative scheduling.

QUESTION BANK

Subject Code: EC1308 Year/ Sem:IV /VII
Subject Name: Embedded Systems Branch : ECE

UNIT-V

Real Time Operating Systems - Part II

Part - A (2 MARKS)

1. Name any two important RTOS.
   1. MicroC/OS II
   2. VxWorks

2. What is sophisticated multitasking embedded system?
   Multitasking provides the fundamental mechanism for an application to control and react to multiple, discrete real-world events. Multitasking creates the appearance of many threads of execution running concurrently when, in fact, the kernel interleaves their execution on the basis of a scheduling algorithm.

3. Explain multi task and their functions in embedded system.
   This system implements cooperative and time-sliced multitasking, provides resource locking and mailbox services, implements an efficient paged memory manager, traps and reports errors, handles interrupts, and auto starts your application at system startup. By following some simple coding practices as shown in the documented coding examples, you can take advantage of these sophisticated features without having to worry about the implementation details.
4. Name any two queue related functions for the inter task communications.
   Queue related functions includes
   5. Creating a queue for an IPC,
   6. Waiting for an IPC messagea t a queue,
   7. Emptying the Queue and Eliminating All the Message pointers,
   8. Sendinga message-pointetor the Queue,
   9. Sending a message pointer and inserting at the Queue Front and
   10. Querying to Find the Message and Enor Information for the Queue ECB.

   To connect a message queue, or create it if it doesn\'t exist. The call to accomplish this is the msgget() system call:
   int msgget(key_t key, int msgflg);

msgget(): returns the message queue ID on success, or -1 on failure (and it sets errno, of course.) The first, key is a system-wide unique identifier describing the queue you want to connect to (or create). Every other process that wants to connect to this queue will have to use the same key.

5. Name any two mailbox related functions.

Function	Parameters returned and passed	When is this OS function called?
OS_Event *OSMboxCreate(void *mboxMsg)	M1 & M1	To create and initialize a mailbox message pointer for the ECB of a mailbox message.
void *OSMboxAccept(OS_EVENT *mboxMsg)	M2 and M2	To check if mailbox message at the *MsgPointer is available at *mboxMsg. If available it returns the pointer and *mboxMsg again points to NULL.
void *OSMboxPend(OS)Event *mboxMsg, unsigned short timeout, unsigned byte *MboxErr)	M3 & M3	To check if mailbox message is pending,. If message is not available it waits, suspends the task and blocks further running. If pending then the task is resumed on availability. Resumes on time out also.

6. Give the function for sending a queue.

Each message is made up of two parts, which are defined in the template structure struct msgbuf, as defined in sys/msg.h:

```
struct msgbuf {
long mtype;
char mtext[1];
};
```

The field mtype is used later when retrieving messages from the queue, and can be set to any positive number. mtext is the data this will be added to the queue

7. Give a function for receiving a message from a queue

A call to msgrcv() that would do it looks something like this:

```
#include
key_t    key;
int msqid;
struct pirate_msgbuf pmb; /* where L\'Olonais is to be kept */

key = ftok(\"/home/beej/somefile\", \'b\');
msqid = msgget(key, 0666 | IPC_CREAT);

msgrcv(msqid, &pmb, sizeof(pmb), 2, 0); /* get him off the queue! */
```

8. Give the steps to destroy a message queue.

There are two ways:

1. Use the Unix command ipcs to get a list of defined message queues, then use the command ipcrm to delete the queue.
2. Write a program to do it for you

9. Give the needs for memory management.

Each new model of computer seems to come with more main memory than the last, but, since the memory requirements of the software rise just as fast, memory is always a precious commodity, hence the need for memory management .

- Memory is allocated to a process when needed
- Memory is deallocated when no longer in use
- Swapping allows the total memory used by all the running processes to exceed main memory
- Virtual memory makes it possible to run a single program that uses more memory than the main memory (normally RAM) available on the system. Virtual memory is normally divided into pages.
- Programs refer to parts of memory using addresses. In a virtual memory system, these are virtual addresses

The virtual address is mapped onto physical addresses by a memory management unit (MMU)

10. Name some application for the VxWorks RTOS.
    1. Automobiles
    2. Avionics
    3. Consumer electronics
    4. Medical devices
    5. Military
    6. Aerospace
    7. Networking

11. What are the various features of VxWorks?
    1. High performance
    2. Host and target based development approach
    3. Supports advanced processor architecture
    4. Hard real-time applications

12. What are the basic functions of VxWorks?
    1. System level functions
    2. Task service functions
    3. Task control functions
    4. IPCs
    5. Network and IO functions

13. What are the task service functions supported by VxWorks?
    1. Task creation and activation distinct states.
    2. Functions for the task creating, running, waiting, suspending and resuming, task pending cum suspending with and without timeouts.

14. What are the different types of semaphores in vxworks? Which is the fastest?

VxWorks supports three types of semaphores. Binary, mutual exclusion, and counting semaphores. Binary is the fastest semaphore.

15. Name any four interrupt service functions supported by VxWorks?

VxWorks runs interrupt service routines (ISRs) in a special context outside of any task's context. Thus, interrupt handling involves no task context switch.

Call	Description
intConnect( )	Connects a C routine to an interrupt vector.
intContext( )	Returns TRUE if called from interrupt level.
intCount( )	Gets the current interrupt nesting depth.
intLevelSet( )	Sets the processor interrupt mask level.
intLock( )	Disables interrupts.
intUnlock( )	Re-enables interrupts.
intVecBaseSet ( )	Sets the vector base address.
intVecBaseGet ( )	Gets the vector base address.
intVecSet( )	Sets an exception vector.
intVecGet( )	Gets an exception vector.

16. What are VxWorks pipes?

Pipes provide an alternative interface to the message queue facility that goes through the VxWorks I/O system. 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.

The call specifies the name of the created pipe, the maximum number of messages that can be queued to it, and the maximum length of each message: status = pipeDevCreate ("/pipe/name", max_msgs, max_length); The created pipe is a normally named I/O device. Tasks can use the standard I/O routines to open, read, and write pipes, and invoke ioctl routines.

17. What is signal servicing function?

VxWorks supports a software signal facility. Signals asynchronously alter the control flow of a task. Any task or ISR can raise a signal for a particular task. The task being signaled immediately suspends its current thread of execution and executes the taskspecified signal handler routine the next time it is scheduled to run. The signal handler executes in the receiving task's context and makes use of that task's stack. The signal handler is invoked even if the task is blocked.

18. Define Micro C/OSII.

Micro C/OSII (commonly termed uC/OSII or mC/OS-Il),i s a low-cost priorty-based preemptive real time multitasking operating system kemel for microprocessors, mainly in the C programming language. It is mainly intended for use in embedded systems.

19. What are the task states in MICRO C/OS-II?

Task states:

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

1. Dormant
2. Ready
3. Running
4. Waiting
5. ISR

20. What are the 2 source files in Micro C/OS-II?

1. Preprocessor dependent source file
2. Preprocessor independent source file

21. What are the basic functions of MUCOS?

- System level: OS initiate, start, system timer set, ISR enter and exit
- Task service function: create, run, suspend, resume
- Task delay
- Memory allocation and partitioning
- IPCs, mailbox and queues

Part – B (16 Marks)

1. Explain detail about Memory allocation related functions
2. Explain briefly about mailbox related functions
3. Explain Queue related functions
4. Explain multiple tasks and their functions
5. Give the steps for creating list of tasks
6. Give the exemplary coding techniques
7. List and explain the various task service functions in VxWorks/MUCOS-II.
8. Explain how IPCs are used in embedded system.
9. Explain the various multiple function calls in Embedded C
10. Write in detail about MUCOS and it's features with a suitable example.
11. Write in detail about VxWorks and it's features with a suitable example.
12. How will you design a application for Automatic Chocolate Vending Machine in detail. (Write with design and code.)
13. How will you design a application for sending Application Layer Byte Stream on a TCP/IP Network in detail. (Write with design and code.)
14. How will you design a application for Adaptive Cruise Control System in Car in detail. (Write with design and code.)
15. How will you design a application to create a Smart Card in detail. (Write with design and code.)
16. Explain about the same case study but more stress on the functions for semaphores and the code for it.
17. How does an RTOS semaphore protect data? Explain by giving example.
18. Draw and explain basic system of an Automatic chocolate vending system
19. Discuss with the diagram task synchronization model for a specific application
20. i) Explain the case study of an embedded system for a smart card.
    ii) Explain the features of VxWorks.
21. Explain the RTOS programming tool MicroC/OS-II.