What is the need for an infinite loop in Embedded systems?

- Infinite Loops are those program constructs where in there is no break statement so as to get out of the loop, it just keeps looping over the statements within the block defined.

Example:

```
While(Boolean True) OR for(;;);
{
//Code
}
```

- Embedded systems need infinite loops for repeatedly processing/monitoring the state of the program. One example could be the case of a program state continuously being checked for any exceptional errors that might just occur during run time such as memory outage or divide by zero etc.,
- For e.g. Customer care Telephone systems where in a pre-recorded audio file is played in case the dialer is put on hold..
- Also circuits being responsible for indicating that a particular component is active/alive during its operation by means of LED's.

How does combination of functions reduce memory requirements in embedded systems?

- The amount of code that has to be dealt with is reduced thus easing the overhead and redundancy is eliminated in case if there is anything common among the functions.

- Memory allocation is another aspect that is optimized and it also makes sense to group a set of functions related in some way as one single unit rather than having them to be dispersed in the whole program.

- In case of interactive systems display of menu list and reading in the choices of user's could be encapsulated as a single unit.

A vast majority of High Performance Embedded systems today use RISC architecture why?

- According to the instruction sets used, computers are normally classified into RISC and CISC. RISC stands for 'Reduced Instruction Set Computing' . The design philosophy of RISC architecture is such that only one instruction is performed on each machine cycle thus taking very less time and speeding up when compared to their CISC counterparts.

- Here the use of registers is optimised as most of the memory access operations are limited to store and load operations.

- Fewer and simple addressing modes, and simple instruction formats leads to greater efficiency, optimisation of compilers, re-organisation of code for better throughput in terms of space and time complexities. All these features make it the choice of architecture in majority of the Embedded systems.

- CISC again have their own advantages and they are preferred whenever the performance and compiler simplification are the issues to be taken care of.

Why do we need virtual device drivers when we have physical device drivers?

Device drivers are basically a set of modules/routines so as to handle a device for which a direct way of communication is not possible through the user's application program and these can be thought of as an interface thus keeping the system small providing for minimalistic of additions of code, if any.
Physical device drivers can't perform all the logical operations needed in a system in cases like IPC, Signals and so on...
The main reason for having virtual device drivers is to mimic the behaviour of certain hardware devices without it actually being present and these could be attributed to the high cost of the devices or the unavailability of such devices.
These basically create an illusion for the users as if they are using the actual hardware and enable them to carryout their simulation results.
Examples could be the use of virtual drivers in case of Network simulators,also the support of virtual device drivers in case a user runs an additional OS in a virtual box kind of a software.

What is the need for DMAC in ES?

- Direct memory access is mainly used to overcome the disadvantages of interrupt and progam controlled I/O.
- DMA modules usually take the control over from the processor and perform the memory operations and this is mainly because to counteract the mismatch in the processing speeds of I/O units and the procesor. This is comparatively faster.

- It is an important part of any embedded systems,and the reason for their use is that they can be used for bursty data transfers instead of single byte approaches.
- It has to wait for the systems resources such as the system bus in case it is already in control of it.

What is Endianness of a system and how do different systems communicate with each other?

- Endianness basically refers to the ordering of the bytes within words or larger bytes of data treated as a single entity.

- When we consider a several bytes of data say for instance 4 bytes of data,XYZQ the lower byte if stored in a Higher address and others in successively decreasing addresses, then it refers to the Big Endian and the vice versa of this refers to Little Endian architecture.

- Intel 80x86 usually follows Little Endian and others like IBM systems follow Big Endian formats.

- If the data is being transmitted care has to be taken so as to know as to which byte,whether the higher or the lower byte is being transmitted.

- Hence a common format prior to communication has to be agreed upon to avoid wrong interpretation/calculations.
- Usually layer modules are written so as to automate these conversion in Operating systems.

How are macros different from inline functions?

- Macros are normally used whenever a set of instructions/tasks have to be repeatedly performed. They are small programs to carryout some predefined actions.

- We normally use the #define directive in case we need to define the values of some constants so in case a change is needed only the value can be changed and is reflected throughout.

```
#define mul(a,b) (a*b)
```

- The major disadvantage of macros is that they are not really functions and the usual error checking and stepping through of the code does not occur.

- Inline functions are expanded whenever it is invoked rather than the control going to the place where the function is defined and avoids all the activities such as saving the return address when a jump is performed. Saves time in case of short codes.

```
inline float add(float a,float b)
{
   return a+b;
}
```

- Inline is just a request to the compiler and it is upto to the compiler whether to substitute the code at the place of invocation or perform a jump based on its performance algorithms.

What could be the reasons for a System to have gone blank and how would you Debug it?

Possible reasons could be:

- PC being overheated.
- Dust having being accumulated all around.
- CPU fans not working properly .
- Faulty power connections.
- Faulty circuit board from where the power is being drawn.
- Support Drivers not having being installed.

Debugging steps which can be taken are:

- Cleaning the system thoroughly and maintaining it in a dust-free environment. Environment that is cool enough and facilitates for easy passage of air should be ideal enough.

- By locating the appropriate support drivers for the system in consideration and having them installed.

Explain interrupt latency and how can we decrease it?

1. Interrupt latency basically refers to the time span an interrupt is generated and it being

serviced by an appropriate routine defined, usually the interrupt handler.
2. External signals, some condition in the program or by the occurrence of some event, these could be the reasons for generation of an interrupt.
3. Interrupts can also be masked so as to ignore them even if an event occurs for which a routine has to be executed.
4. Following steps could be followed to reduce the latency
- ISRs being simple and short.
- Interrupts being serviced immediately
- Avoiding those instructions that increase the latency period.
- Also by prioritizing interrupts over threads.
- Avoiding use of inappropriate APIs.

How to create a child process in linux?

- Prototype of the function used to create a child process is pid_t fork(void);
- Fork is the system call that is used to create a child process. It takes no arguments and returns a value of type pid_t.
- If the function succeeds it returns the pid of the child process created to its parent and child receives a zero value indicating its successful creation.
- On failure, a -1 will be returned in the parent's context, no child process will be created, and errno will be set.
- The child process normally performs all its operations in its parents context but each process independently of one nother and also inherits some of the important attributes from it such as UID, current directory, root directory and so on.

Significance of watchdog timer in Embedded Systems.

- Watchdog timer is basically a timing device that is set for predefined time interval and some event should occur during that time interval else the device generates a time out signal.

- One application where it is most widely used is when the mobile phone hangs and no activity takes place, in those cases watchdog timer performs a restart of the system and comes to the rescue of the users.

- It is used to reset to the original state whenever some inappropriate events take place such as too many commands being given at the same time or other activities that result in malfunctioning of the GUI. It is usually operated by counter devices.

If you buy some RTOS, what are the features you look for in?

- Deterministic operating system having guaranteed worst-case interrupt latency and context-switch times.
- Documentation providing for the minimum, average, and maximum number of clock cycles required by each system call.
- Interrupt response times should be very minute.
- Context switch time should be very low.
- Compatibility with several plugin devices.
- Overall it should be very reliable.

Why is java mostly used in embedded systems?

- Java was mainly designed and conceptualised for code that can work on different platforms without any hassles and also for being secure enough so as to not harm or corrupt other modules of code.

- Features like exception handling, simple syntax and Automatic Garbage collection all work in its favour as the language for use in ES's.

- Also that it is widely used in the form of Java applets makes it very popular confining it to the limits of JVM. It is Dynamic in nature.

- Its use is also being exploited in enterprise systems in the form of J2EE, J2SE, J2ME in case of mobile applications.

Differentiate between mutexes vs semaphores.

-Semaphores is a synchronization tool to overcome the critical section problem.

- A semaphore S is basically an integer variable that apart from initialization is accesses only through atomic operations such as wait() and signal().

- Semaphore object basically acts as a counter to monitor the number of threads accessing a resource.

- Mutex is also a tool that is used to provide deadlock free mutual exclusion. It protects access to every critical data item. If the data is locked and is in use, it either waits for the thread to finish or awakened to release the lock from its inactive state.

What are the commonly found errors in Embedded Systems?

- Damage of memory devices due to transient current and static discharges.
- Malfunctioning of address lines due to a short in the circuit.
- Malfunctioning of Data lines.
- Some memory locations being inaccessible in storage due to garbage or errors.
- Improper insertion of Memory devices into the memory slots.
- Faulty control signals.

What is the need for having multibyte data input and output buffers in case of device ports?

- It's normally the case that some devices transfer the output either in a bursty or a sequential manner and also during input entry. If we take the example of keyboards, all the data entered is stored in a buffer and given at a time or one character at a time.

- In case of networking there may be several requests to access the same resource and all these are queued in a buffer and serviced in the order they are received. Hence to avoid the input/output units from getting overloaded with requests, we use multibyte buffers.

What is the difference between Hardware design and Software Design?

- Hardware design is designed with the collaboration of interconnected parallel components that inherits the properties of each other. Whereas, Software design is designed with sequential components, that are based on objects and threads.

- Hardware design structure doesn’t change dynamically and it can’t be created, modified or removed easily. Whereas, Software design structure can be changed dynamically and re-usability features, used to define the components. It also includes easy creation, modification and removal of the components from the software.

- Hardware design focuses on individual components that are represented using analytical model that uses the transfer functions. Whereas, Software design represent the components using computation model that can have abstract execution engine or it can use the virtual machine that are non-deterministic.

What are the differences between analytical and computational modeling?

- Analytical model allows the components to deal with the concurrency that are given during the process and the quantitative constraints that might come in between the components. Whereas, computational model deal with the non-deterministic abstraction hierarchy that has computational complexity to deal with the concurrency and allow it put also the physical constraints.

- Analytical models can’t deal with the partial and incremental specifications that are non-deterministic. It is also not good in controlling the computation complexity that is used in the hardware design. Whereas, Computational model can, deal with constraints easily and it provides an upgradeable solution.

- Analytical model is the equation based model that doesn’t have the time-sharing and parallelism concepts. Whereas, time-sharing and parallelism is used, in the abstract method that provides the theories of complexity and the real time evaluation.

What are the functional requirements that are used in the embedded systems?

Functional requirements specifies the discrete and the logic to provide the services, functionality, features and the implementation that is independent from the components that are getting used in it. These are used to represent the constraints that are in the form of physical and define the probability to specify the components discretely from each other. The functional requirements are given for the hardware as well that gives more performance and measures the physical resources that are present like clock frequency, latency, etc. Functional requirements allow the system and hardware machines to transfer the functions with the non-deterministic probability.

Why is Model transformations used in the embedded system?

Model transformations involve multiple models that are used to define different views of a system. It provides different level of granularity that it doesn't use either the top-down approach or the bottom-up approach to implement the basic functionality of the system. It is used to integrate the library components used that involves the iteration of the model that needs to be constructed. It also involves the analysis of the model so that the process can be made automated by using the construction tools. The compilation made the progress by improving the code that is written in high level language and the code generator produce the code that is required for the machine language.

What is interaction semantics used in embedded systems?

Interaction semantics allow the actions to be performed by the system components to allow it to get the global behavior. The interaction can be atomic or non-atomic dependent on the interaction between the components. These components can't be modified using the interference having the other interactions. Languages that are used, having buffered communication, and other languages, that include multi-threaded languages that use non-atomic interactions. There are two types of interactions that are used:
- Strong synchronization: allow the components to participate together and have strong bonding in between.

- Weakly synchronizing: are asymmetric that required the communication from both the objects.

What are the different types of system involved in embedded system?

Embedded systems are used to give the response in real time. So, it consists of the real time systems that allow the correct information to be passed to get the correct responses. For example, it includes of the flight control system that produce the responses in real time and it always the take the values also in real time. If any delay been caused by the system then it deals in the fatal error. The real time system includes the system that provides the response on time with the small delay. Real time systems include of many other system such as:

Hard Real-Time Systems - These are the systems that server the purpose of having constraints that are hard and it totally depends on the time to provide the response on time.

Soft Real-Time Systems - These systems serves the purpose of having few delays in giving up the responses that can tolerate small variations.

Hybrid Real-Time Systems - These systems includes the properties from both the systems and increases the performance.

What are the different types of Buses used by the embedded systems?

The buses are used to pass the messages between different components of the system. There are buses existing as:

- Memory Bus: it is related to the processor that is connected to the memory (RAM) using the data bus. This bus includes the collection of wires that are in series and runs parallel to each other to send the data from memory to the processor and vice versa.

- Multiplexed Address/Data Bus: Multiplex data bus consists of the bus that can read and write in the memory but it decreases the performance due to the time consumed in reading and writing of the data in the memory.

- De-multiplexed Bus: these consists of two wires in the same bus, where one wire consists of the address that need to be passed and the other one consists of the data that need to be passed from one to another. This is a faster method compared to other.

- Input/Output bus: it uses the multiplexing techniques to multiplex the same bus input and output signals. This creates the problem of having the deadlock due to slow processing of it.

What is the main function of Multiplexed Address/Data Bus?

The memory bus is used to carry the address and the data from the processor to the memory so that it can be easily accessed by the devices. These buses carry the value of the data that has to be passed for the proper functioning. The use of the technique “Time division multiplexing” is used that allow the reading and writing of the data to be done from the same bus line. This requires lots of time to be given to the bus so that it can complete the read and write operation of the data in the memory. This is very expensive process due to the data transfer technique that is used in between the processor and the memory. This also gives the concept of cache and provides algorithms to solve the problems occurring in read and writes operations.

How does Input/Output bus functions?

Input and output devices or functions allow the user to interact with the external files. Input and output functions are used to transfer the load on the bus. It uses the multiplexes having the input and output signals that remain same. Input and output buses move at the slower rate or speed than the processor speed. This increases the problem of bottleneck or the deadlock due to poor performance. There is a possibility to send more transistors for a layout to be given. These different devices may have very different speeds of communication. When programming IO bus control, make sure to take this into account.

In some systems, memory mapped IO is used. In this scheme, the hardware reads its IO from predefined memory addresses instead of over a special bus. This means you'll have simpler software, but it also means main memory will get more access requests.

What is the function of simple thread poll in embedded system?

Simple thread poll allow the ready output to be passed for checking by giving it to the bus that is free and then the output is sent along the thread. The bus can send the output depending on the time that has been given and during the transfer the user won’t be able to perform any other operation. The input is given after finding out the bus is free or not and if it free then a check is made to see that the input exists or not. This thread poll is easy to understand but it is not efficient method to allow the data to be put over the bus manually. The problem of not doing multi-tasking can occur due to doing one task at a time. The method is only be used when input/output occurs at interval that are infrequent.

Why is it better to use multi-threading polling then single threading model?

Multi-threading allows a simple thread to be stored and polled. There is no Input/output function that is applied when it is having the poll. When there is no poll available to spawn it makes the system to sleep for an amount of time till the request for another poll reaches. If there is one process that is running then it divides that process into multiple threads and processes it accordingly. It allows the main thread to process all the request and produce the output by combining all other. Multi-threading allows the main thread not to put off the result or the output that will be generated. It also allow the priority of the thread to be changed by allowing to set the priority of the input/output process. It also has some problems with the polling interval that can make a thread starve for some time if the request isn’t handled properly.

How does the interrupt architecture works?

Interrupt architecture allows the use of interrupt by the processor whenever an Input/output is ready for the processing. The processor in this case calls a special function to handle the request that comes and leave all the work that is getting performed at that time. The special function that is known as interrupt handler or the interrupt service routine consists of all the input, and output queries, or the interrupts handled by it. It is an efficient and simple way to handle the interrupts. It uses only one function to deal with the interrupts. There are properties of starvation that can creep in when handling the input/output requests. The data can be lost if the interrupt doesn't get handled before the time runs out. This is a technique that is use to deal with the short processes that involve input and output.

How does the interrupts handle by using the threads?

The interrupts that comes in between the input/output operations gets detected when the input/output devices are ready. The interrupt never gets handled directly rather, it sends the interrupt signal to the thread to the input/output device that is ready to allow the thread to take necessary actions. The thread uses the signaling concept that allows the initialization to be done using the semaphore that keeps the states updated and handle the interrupt in an easy way. The input/output device getting the request and it also passes the semaphore to handle. The input/output device takes the semaphore that is ready. The thread is having the minimum latency that uses the first level interrupt handler to handle the interrupts completely. It allows the priority of the thread to be set and it doesn't allow the context to be change as well.

What is the function of DMA controlled in embedded system?

DMA stands for Direct Memory Access controller that handles the allocation of the memory dynamically to the components and allows the data to be transferred between the devices. It is used for the communication between different input/output devices. It automatically detects the devices that are present for the transfer of data between the input/output devices. The interrupt can be used to complete the data transfer between the devices. It is used to give the high quality performance as, the input/output device can perform the operations that are in parallel with the code that are in execution phase. It can’t be used in all the systems. It consists of 8-bit microcontrollers that are used to control the overall system in execution.

What are the different types of customizations that is used with the “volatile” keyword?

Volatile keyword is used to show that the value can be changed anytime in the program. It is used for the compiler purpose and for the customization that works with the normal variables that are stored in the memory. There are three types of optimizations associated with the “volatile” keyword:

- "Read" optimizations: allow the variable to be read once and put it in the register. If it is done then there is no re-reading of the variable during each and every time the program is compiled. The value can be used from the cache that is present in the register.

- "Write" optimizations: allow the variable to be written such that the last write of the variable will be considered and it will be processed on. This takes the normal values that are stored in the memory.

- Instruction reordering: allow to reorder the instructions that are used by the compiler and if any modification are required after being written once. The registers are used to perform the task and keep everything together.

Write a program to show the functionality of Power-save super loop.

To check the loop time of the program the power-save super loop is used. If the average loop time of the program is 1ms, and it requires only few instructions to be checked every second the program will save the state and build a delay that will be caused to read the input on every loop and it saves lot of energy or the power that needs to be used. The function that is required to be performed to show the functionality is:

```
Main_Function()
Function
{
  Initialization();
  Do_Forever
  {
    Check_Status();
    Do_Calculations();
    Output_Response();
    Delay_For_Next_Loop();
  }
}
```

Why does pre-emptive multi-threading used to solve the central controller problem?

Multi-threading provide lot of functionality to the system to allow more than one task can be run at a time. It allows a process to execute faster with less difficulty. But, if there any problem comes in any program or the process than the entire system comes to a halt and slows down the whole system. To control the behavior of this the preemptive multi-threading is used. The control in this case is being shifted from one process to another at any time according to the requirement provided. It allows the program to give the control to another program that is having the higher priority. It includes of many problems like giving of a control by a process half way through in execution and the preemption of the process takes place then the data will be entered as corrupted in the memory location, multi-threading keeps the synchronization that is to be performed between different components of the system and the program and try to avoid the problem mentioned above.

What are the rules followed by Mutexes?

Mutex is also called as Mutual Exclusion is a mechanism that is used to show the preemptive environment and allow providing security methods like preventing an unauthorized access to the resources that are getting used in the system. There are several rules that has to be followed to ensure the security policies:

- Mutex are directly managed by the system kernel that provides a secure environment to allow only the applications that passes the security rules and regulations. The mutex consists of objects that are allowed to be called by the kernel.

- Mutex can have only one process at a time in its area that is owned by the process using it. This allows less conflict between the different applications or processes that wait for their turn to execute it in the kernel area.

- Mutex can be allocated to another mutex that is running some task at a particular time and allow the kernel to have synchronization in between them.

- If Mutex is allocated to some other process then the area will consist of the process till the area is having the process in it.

What is the purpose of using critical sections?

Critical section allows the process to run in an area that is defined by that process only. It is a sequence of instructions that can be corrupted if any other process tries to interrupt it. This process allow the operating system to give the synchronization objects that are used to monitor the processes that are up and running so that no other process will get executed till the critical region consists of a process that is already running. The example includes removal of the data from a queue running in a critical section and if not protected then it can be interrupted and the data have chances of getting corrupted. The processes exit from the critical section as soon as they finish the execution so that the chances can be given to other processes that are waiting for their chance to come.

What is the function of Watchdog timer in embedded system?

The embedded system should have a function that can allow the fixing the system if anything goes wrong with it. Watchdog timer is a tool that is used in embedded system and having a long-fuse that runs several seconds. Watchdog timer includes the automated timing control that count down the number from max to 0 and when the counter reaches the zero, this WDT reset the micro-controller that gets turned off when the timer was in initial phases. Watchdog require the user to put some value before it runs out of time and reset the whole process as this can harm the data and the system. Resetting by WDT can be done when the process is half complete. Watchdog timer have the counter that is used to watch the processes that are running and if there is any issue occurs then WDT itself times out. The resetting of the system will be done to always give it the best possible way to execute the process.