Embedded Interview Questions and Answers

Introduction

The continuous investments in cloud computing and artificial intelligence are expected to drive significant growth in the embedded market. Thus, it creates a promising career path for freshers and experienced professionals. The top **embedded interview questions and answers** are provided here for the benefit of all candidates. Learn more with our Embedded training in Chennai.

List of Embedded Interview Questions for Freshers

1. What is an embedded system?
2. List some real-time embedded systems
3. Define microcontroller.
4. Why does an embedded system require an infinite loop?
5. What is a semaphore?

6. What code does the startup use?
7. When should a volatile keyword be used?
8. What is ISR's complete form?
9. What are the uses of a timer in an embedded system?
10. What is a Watchdog timer?
11. Explain a recursive function.
12. What is an embedded automotive system?
13. Define memory leaks.

Ready to move from theory to practice? Follow our Embedded Systems Tutorial for Beginners to build your first application from scratch.

Embedded Interview Questions and Answers for Freshers

1. What is an embedded system?

An embedded system is hardware with software that performs a specialized job or function. It may be part of a larger or full system. They may have fixed functionality or be programmable.

Automobiles, mobile devices, consumer electronics, agricultural and processing industry equipment, industrial machinery, and so on are a few examples of embedded systems.

2. List some real-time embedded systems.

Real-time embedded systems are computer programs to monitor, react to, or manage an external environment. This environment and the computer system are connected through actuators, sensors, and input-output interfaces.

3. Define microcontroller.

A microcontroller is a tiny integrated circuit that manages a single function in an embedded system. A microcontroller is a single chip with memory, input/output (I/O) peripherals, and a CPU. One self-sufficient system that can be utilized as an embedded system is a microcontroller.

4. Why does an embedded system require an infinite loop?

Embedded systems need infinite loops for repetitive processing or condition monitoring. Consider the case of a program that is continuously scanned for any odd errors that can arise during runtime, such as dividing by zero or a memory blackout, etc.

5. What is a semaphore?

It is a non-negative shared variable between threads. It accomplishes process synchronization and resolves the crucial selection issue. Binary and counting semaphores are the two types of semaphores.

6. What code does the startup use?

This code is called before the main function is executed. It sets up a system on which an application can operate. We refer to it as an assembly language.

7. When should a volatile keyword be used?

A volatile keyword is used when a compiler behaves unexpectedly after optimization.

8. What is ISR's complete form?

The term "*Interrupt Service Routines*" is ISR. It is applied in the event of a disruption. The software stores these processes in a memory location.

9. What are the uses of a timer in an embedded system?

Timer applications in embedded systems include the following:

- It serves as the system's RTC (Real-Time Clock).
- It starts an event after the specified amount of time.
- It records an event's count value.
- It calculates the time between any two occurrences.
- It shortens the time spent on different procedures and jobs.
- It uses RTOS to schedule tasks.

10. What is a Watchdog timer?

When something goes wrong with the system, a watchdog timer is an electronic device designed to perform a specific function after a predetermined time.

11. Explain a recursive function.

Recursion is the process of repeating an object in a self-similar way. When a program allows you to call a function inside another function, this is known as a recursive call of the function.

12. What is an embedded automotive system?

A computer system that can function as a control system for electronic devices that have an impact on an automobile's data or mechanism is called an automotive embedded system.

13. Define memory leaks.

A memory leak is a type of resource leak that happens when computer software allocates memory improperly, making it impossible for memory that is no longer needed to be released. A memory leak occurs when something is stored in memory but is not accessible by the code that is now running.

Explore our embedded system course syllabus to get started.

List of Embedded C Interview Questions

1. What is an embedded C?
2. In embedded C, what leads to a segmentation fault error?
3. In embedded C, what is the concatenation operator?
4. Can a static variable be declared in a header file?
5. Explain the reentrant function.

Embedded C Interview Questions and Answers

1. What is an embedded C?

An expanded form of the C programming language is called Embedded C. It can be utilized to create microcontroller-based applications such as device drivers (e.g., WiFi, cameras, etc.).

2. In embedded C, what leads to a segmentation fault error?

In embedded C, a segmentation fault error is a runtime problem that can arise for several reasons.

- It is possible that a pointer is not pointing to the correct memory location or address.
- If a user attempts to access a read-only memory location.
- If the user attempts to utilize a pointer to release memory, it means it has already been released.

3. In embedded C, what is the concatenation operator?

The use of ## designates the concatenation operator. It is utilized in macros to concatenate the macro's arguments. It is important to remember that the values of the arguments are not concatenated; just the arguments themselves are.

Example Code:

```
#define CUSTOM_MACRO(x, y) x##y

main(){

 int xValue = 20;

 printf("%d", CUSTOM_MACRO(x, Value));    //Prints 20

}
```

This is how we might conceptualize it: The macro just returns xy -> is the concatenation of x and y if arguments x and y are given.

4. Can a static variable be declared in a header file?

Static variables are local to the block in which they are defined, have a single initialization, and remain such until the program terminates. Definitions are needed for declarations of static variables. A header file may contain its own definition.

However, if we do this, each source file in which the header is included will contain a private duplicate of the header file's variable. Using static variables in a header file is not advised because it is not preferred.

5. Explain the reentrant function.

Reentrant functions are those that can be securely paused mid-execution and then safely called again, or re-entered, to finish the execution. Events or signals from the outside world or internal signals, like calls or jumps, can cause an interruption. The reentrant function picks up where the last execution left off and continues through completion. Explore Embedded system project ideas to practice more.

List of Embedded Interview Questions for Experienced

1. How does writing less-than-optimal code result from utilizing the address of a local variable?
2. In what way might a combination of functions lower the memory requirements in embedded systems?
3. In embedded C, what is virtual memory, and how might one be implemented?
4. How can interrupt latency be reduced, and what causes it?
5. To determine if a given integer is a power of two or not, write a program.

Embedded Interview Questions and Answers for Experienced

1. How does writing less-than-optimal code result from utilizing the address of a local variable?

The most efficient optimization performed by the compiler is register allocation. In other words, rather than using memory first, it manipulates the variable using the register. Registers are frequently used to assign local variables. Nevertheless, if we take the address of a register, the compiler will not assign a local variable to it.

2. In what way might a combination of functions lower the memory requirements in embedded systems?

If there is any overlap between the functions, redundancy is eliminated and less code needs to be managed, which lowers overhead.

Memory allocation is another optimized feature. It makes sense to group functions that are connected into a single unit instead of dispersing them throughout the program.

3. In embedded C, what is virtual memory, and how might one be implemented?

If there is not enough physical memory, virtual memory can be used to automatically allocate storage to the processes. The ability to have more virtual memory than physical memory is the primary benefit of employing virtual memory. It can be put into practice by employing the paging approach.

Here's how paging operates:

- A lazy swapper known as a pager is used to swap a process into memory whenever it needs to be executed.
- Using a predetermined algorithm, the pager attempts to determine which page requires memory access and switches that process.

This makes sure that only the steps that are required are swapped using pages, and that the entire process is not swapped into memory.

- This lowers the amount of physical memory needed and cuts down on the time needed for swapping and pointless memory page reading.

4. How can interrupt latency be reduced, and what causes it?

The different reasons for interrupt latency are as follows:

- **Hardware:** The signal needs to be in sync with the CPU clock cycles every time an interrupt happens.

Before the interrupt signal reaches the processor for processing, it may take up to three CPU cycles, depending on the processor's hardware and synchronization logic.

- **Pipeline:** Instructions are pipelined on the majority of contemporary CPUs. Once an instruction has advanced to the last pipeline stage, execution takes place.

It would take a few more CPU cycles to add new instructions to the pipeline after an instruction has completed its execution. This makes the latency worse.

Making sure that the ISR procedures are brief will help to decrease interrupt latency. The lower-priority interrupt would be delayed and cause more latency if it were to occur while the higher-priority interrupt was still being processed.

Smaller ISR routines for lower-priority interrupts would be helpful in these situations to cut down on latency. Additionally, the CPU's improved scheduling and synchronization algorithms would aid in reducing the ISR latency.

5. To determine if a given integer is a power of two or not, write a program.

Bitwise operators can be used to do this.

```
void main (){
  int num;
  printf ("Enter any no:");
  scanf ("%d", &num);
  if (num && ((num & num-1) == 0))
```

```
        printf ("Number is a power of 2");
    else
        printf ("Number is not a power of 2");
}
```

Conclusion

We hope that these **embedded interview questions and answers** will enable you to respond to interviews with more confidence. Hone your skills by enrolling in our software training institute in Chennai.