Please use this document as examples of common embedded engineering concepts. We hope this resource will help prepare your problem solving skills for the interview process. However, these are not sample questions that may be asked in the interview.

Embedded Software Concepts

Preprocessor

- Using the #define statement, how would you declare a manifest constant that returns the number of seconds in a year?
    - Consider:
        - What if this was used on a processor with an short (16 bit) integer type?
        - Is a constant computed from several values less efficient at run time than a simple constant?

- Write the “standard” MIN macro-that is, a macro that takes 2 arguments and returns the smaller of the 2 arguments.
    - Consider:
        - What arguments to the macro could lead to an unexpected result or side effect?
        - What alternatives does C++ offer to construct an equivalent to this MIN macro? Do they share the same potential problems?

- What is the purpose of the preprocessor directive #error?
    - Consider:
        - How could it be useful in practice?
        - When would using this be better than a run-time error check?

Data Declarations

- Using variable a, give definitions for the following:
    - An integer
    - A pointer to an integer
    - A pointer to a pointer to an integer
    - An array of 10 integers
    - An array of 10 pointers to integers
    - A pointer to an array of 10 integers
    - A pointer to a function that takes an integer as an argument and returns an integer
    - An array of 10 pointers to functions that take an integer argument and return an integer

---

Bit Manipulation

- Embedded Systems always require the user to manipulate bits in registers or variables. Given an integer variable a, write two code fragments.
    - The first should set bit 3 of a.
    - The second should clear bit 3 of a.
    - In both cases, the remaining bits should be unmodified.
- Consider:
    - What it the clearest way to do this?
    - What issues could be caused when this code is maintained?

Accessing fixed memory locations

- Embedded systems are often characterized by requiring the programmer to access a specific memory location. On a certain project, it is required to set an integer variable at the absolute address 0x67a9 to the value 0xaa55. The compiler is a pure ANSI compiler. Write code to accomplish this task.
- Consider:
    - Why would we want to do this at all?
    - This seems relevant to device peripheral access, but what else might be needed to give a reliable implementation?

Interrupts

- Interrupts are an important part of embedded systems. Consequently, many compiler vendors offer an extension to standard C to support interrupts. Typically, this new keyword is called _interrupt. The following code uses _interrupt to define an interrupt service routine (ISR). Comment on the code:

```
_interrupt  double  compute_area
(double radius)
(
   double area = PI * radius *
  radius;
printf("\nArea = %f",area);
return area;
)
```

Dynamic Memory Allocation

- Although not as common as in non-embedded computers, embedded systems do still dynamically allocate memory from the heap. What are the problems with dynamic memory allocation in embedded systems?

Typedef

- Typedef is frequently used in C to declare synonyms for pre-existing data types. It is also to use the preprocessor to do something similar. For instance, consider the following code fragment:

```
#define dPS struct s *
typedef struct s* tPS;
```

The intent in both cases is to define dPS and tPS to be pointers to structure s. Which method, if any, is preferred and why?

Alternate I/O access question.

Embedded code often involves interacting with hardware via memory accesses. Modern processor architectures and compilers include features to improve performance; but those features often interact with code that accesses memory in complex ways.

Consider:

- What actions occur during an I/O access and where?
- What compiler features impact code designed to perform peripheral memory access?
- Likewise how does the presence of modern processor features such as caches and out-of-order impact code interacting with hardware?

Alternate DMA question.

High performance peripherals, such as Ethernet controllers, often use DMA to transfer large amounts of data efficiently. Ensuring the processor and memory are consistent may require the additional software or hardware operations.

Consider:

- What are the steps of a DMA transfer?
- How is the processor cache kept consistent with memory? (There are both hardware and software methods).
- DMA transfer is very efficient, but it could still affect performance of unrelated code. How?

Alternate Interrupt question.

Peripheral devices often have access to processor interrupts to get the attention of the processor when certain events happen.

Consider:

- What happens during an interrupt operation?
- It's often simpler to put a significant amount of code in the interrupt handler; but this can lead to poor results in practice.
    - What context does an interrupt handler run in?
    - How would you get access to data you need in the interrupt handler (especially if multiple interrupts are serviced by the same handler)?
    - How would having a long-running interrupt handler adversely affect other code?
    - How would you minimize the amount of processing in an interrupt handler?
    - Peripherals often generate a very large number of interrupts. What techniques could you use to reduce the number of interrupts seen by the processor?

Architecture Design

OTA - Mechanics of an over the air update of a device flash, and dealing with all failure contingencies.

Compiler Qualifiers

- C has several qualifiers - static, const, and volatile. Each qualifier can be generally be used in several different ways.
- Consider:
    - What does each qualifier use tell the compiler?
    - What effect do you expect it to have during compilation?
    - What effect do you expect it to have on the generated code?
    - What benefit does it confer to the source code?

***Compilers specific:

Obscure Syntax

- C allows some appalling constructs. Is this construct legal, and if so, what does this code do?

Int, a = 5, b = 7, c;
c = a+++b;