Toptal

# 10 Essential Embedded Software Engineering Interview Questions *

Toptal sourced essential questions that the best embedded software engineers can answer. Driven from our community, we encourage experts to submit questions and offer feedback.

Submit an Interview Question

## Describe the pros and cons of using a generic real-time operating system (RTOS) on a mid-range microcontroller.

Hide answer

RTOSes can significantly ease the development of complex products, which can translate into faster development cycles. They often support compartmentalizing code into tasks, implement cross-task communication mechanisms, and commonly include abstractions (“drivers”) for platform-specific hardware, which makes porting firmware to new hardware easier. Because of all that, they also introduce overhead in code size and CPU usage, which is not acceptable for

## What are some common issues when handling interrupts?

Hide answer

the device and application—and this limits the complexity of what can be done in their code. Also, the context in which the interrupt handler code is executed can, for either hardware or software reasons, prevent the usage from within the interrupt handler code of:

- Common library functions
- Access to peripherals and devices
- Even certain types of CPU instructions

The usual way to mitigate this is to have the interrupt controller set a special variable which is observed by non-interrupt code, and which can then perform arbitrary actions

## In platforms with significant constraints on memory size, is it more preferable to allocate memory statically or dynamically?

Hide answer

It's preferable to use static memory allocation on platforms with memory sizes in the low kilobytes and below. This is because data overhead, CPU overhead, and memory fragmentation can be significant issues when using dynamic memory allocation.



## Why are C and C++ still very popular and widely supported in embedded firmware development?

Toptal

Hardware constraints, both for memory sizes and CPU speed, limit what can be done on embedded devices. C and C++ usually have very minimal overhead and are very “close to the hardware” in terms of abstractions offered to developers. This makes them suitable for even the smallest devices.

## How many wires are required to reliably implement TTL-like serial communication between two devices, and why?

Hide answer

TTL-like serial communication is often used to interface small microcontroller-based devices to larger computer systems, either for general communication or for uploading firmware. This type of communication uses two wires, one for each direction, called TX (transmit) and RX (receive.) But there also needs to be a common electrical ground level shared between the devices, so the minimum number of wires to reliably implement TTL serial communication is three. (The requirement for common electrical ground is also present in I2C and SPI.)

## Since 32-bit and 64-bit microcontrollers exist, why are 8-bit ones still in use?

Hide answer

The general reason is picking the right tool for the job. The three most common reasons are backwards compatibility, price, and electrical power consumption. Backwards compatibility is important when interfacing with existing infrastructure, especially in industrial environments, where in many cases, the electrical and operational constraints impact the choice of microcontrollers.

Generally, smaller microcontrollers (with narrower primary registers) are also cheaper. But they can contain a very large selection of peripherals and interfacing options, so they can be used in many situations that require advanced functionality but not high CPU speed.

Smaller microcontrollers also generally require less power to operate, which is especially

Hide answer

No. Unless the microcontroller is specially constructed to offer countermeasures against firmware downloading and/or modification, any code and data uploaded to a microcontroller should be considered relatively easy to download and modify. (Such hardened microcontrollers are usually expensive.)

## Describe the role of a watchdog timer.

Hide answer

A watchdog timer is a feature of many microcontrollers—usually implemented with specific dedicated hardware—that can be used to check whether the software running on the microcontroller hung.

Microcontrollers are designed to be sturdy and resilient. But there's still any number of issues that can affect hardware stability. There can also be an unhandled combination of events on the software side. Both of these can cause microcontrollers to "hang," either electrically or in an infinite loop in software.

A watchdog timer is a subsystem which needs to be explicitly notified by the software that everything is running as expected, within a specific amount of time. If the watchdog does not receive the notification it expects, it will perform some action, such as resetting the

## What are the most important characteristics of UART-based (also called *RS-232-like* and *TTL-like*) serial communication, I2C communication, and SPI communication?

Hide answer

Simple UART-based serial communication—with or without UART hardware—is the least demanding communications protocol to implement, but comes with severe limitations:

devices.

- It's most commonly used at slow bit rates (up to 115,200 bps).

I2C can connect up to 127 devices on the same electrical bus, and each device is individually addressable. One of the devices, a master device, generates a clock signal shared by all the others, called slave devices. There is only one data wire, so all communication is unidirectional. (It's commonly used to communicate with sensors on a PCB, which often use simple request-response protocols.)

The SPI bus is designed for fast, bidirectional communication with complex devices, which can involve cases such as transferring a large volume of data in bulk. With SPI, all devices share the

## Discuss a couple of options for wireless communication between embedded devices.

Hide answer

On the high end of cost and complexity, wireless communication can be implemented using one of the wifi standards. These offer great bandwidth, are interoperable with many other devices, and can be long-range. But wifi standards are also fairly complex and require dedicated hardware.

Bluetooth is a reasonable choice for interfacing between different types of hardware over short distances, i.e., those of up to 15 yards (~14 meters). It also requires specialized hardware, but such hardware is usually cheap and simple to use. Devices using Bluetooth for communication often emulate a serial line between them.

There are also custom radio-based communication devices and protocols which work on the same frequencies as WiFi and Bluetooth (around 2.4 GHz), but with simpler protocols that are incompatible with the standard ones. They are usually cheaper than Bluetooth and simpler to implement.

There are also transceivers operating at low frequencies such as 433 MHz. While they offer very low bitrates over short distances—up to 10 yards (~9 meters)—they are extremely cheap and easy to implement. If larger distances are required and a small bitrate is acceptable, which

Toptal®

## Submit an interview question

Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Name *

Email *

Enter Your Question Here ... *

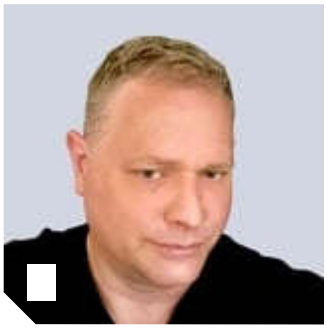
Enter Your Answer Here ... *

Toptal

* I agree with the Terms and Conditions of Toptal, LLC's Privacy Policy

* All fields are required

Submit a Question



## Byron Formwalt

Freelance Embedded Software Engineer

United States Toptal Member Since March 24, 2014

Byron has 22 years of experience in highly technical algorithm development. He has a Ph.D. in electrical engineering and a wealth of experience in artificial intelligence, machine learning, and video/image... Show More

View Byron

Embedded Software Engineering | C | C++ | Ruby | Ruby on Rails (RoR) | iOS SDK | Xcode | iOS | Linux

## Erlend Hamberg

Freelance Embedded Software Engineer

Norway Toptal Member Since May 11, 2015

Erlend has a large range of experience in software development, having worked on projects from kernel driver development to web apps. He is most proficient on the back-end side of things and knows how important... Show More

View Erlend

Embedded Software Engineering | Haskell | Git | Linux | C

## Guri Labartyava

Freelance Embedded Software Engineer

With a bachelor's degree in computer science and math and a master's in computer engineering, Guri possesses a solid academic background along with several years of experience, working on complex projects on...
Show More

View Guri

Embedded Software Engineering | C | C++

## Looking for Embedded Software Engineers?

Looking for Embedded Software Engineers? Check out Toptal's embedded software engineers.

## Toptal Connects the Top 3% of Freelance Talent All Over The World.

Join the Toptal community.

Learn more

Skills in High Demand by Clients

Shopify

About

Top 3%

Toptal®

SharePoint

React.js

Vue.js

Ultimate Freelancing Guide

Blog

Community

About Us

Contact

Contact Us

Press Center

Careers

FAQ

Social

®

The World's Top Talent, On Demand ™