Embedded Systems Essentials with Arm: Getting Started

Module 1

KV1 (1): Introduction to Embedded Systems

An embedded system is a specialized computing system with hardware that is based around a chip (this could be a microcontroller or microprocessor). Its embedded hardware will vary depending on what the system is designed to do.

There's no ultimate set of operational capabilities for an embedded system, but there are a few things we can expect.

Embedded systems usually have hybrid processing and computation capabilities.

They are typically designed to perform a well-defined task which repeats either periodically or spontaneously upon a trigger. These task-specialized designs might interact with larger systems, and the surrounding environment.

Finally, embedded systems have a low power budget and relatively high performance, and are low cost.

Embedded systems are often subject to real-time computing constraints.

A real-time system is a system that guarantees a meaningful output is produced within a defined short period of time or interval.

This means embedded systems must respond quickly to sequences and combinations of events, and typically must perform multiple separate activities at the same time.

Furthermore, because many embedded systems must operate independently for long periods of time, they must handle faults without crashing. This often means that the fault-handling code is larger and more complex than the normal-case code.

Consequently, embedded systems perform diagnostic functions to help service personnel determine problems quickly.

Now, let's look at the differences between a CPU, an MCU, and an embedded system.

The microprocessor, also known as a central processing unit or CPU, is the central core of an embedded system. It is a multipurpose, clock driven, programmable electronic device designed to perform arithmetic and logic operations using an arithmetic logic unit, or ALU. It usually performs the instruction cycle of fetching, decoding and executing.

The microprocessor also communicates with other external components with a memory unit via a memory interface, and can perform I/O operations based on specific instructions.

A microcontroller unit or MCU consists of a microprocessor plus additional peripheral components such as memory blocks, digital I/Os, analog I/Os, timers, and other basic peripherals.

The microprocessor is usually a single-core CPU to optimize power and cost efficiency. On-chip memory components such as random-access memory and read-only memory are also optimized for speed, cost, durability, and energy efficiency.

Additional hardware components can include a variety of input and output components, timers, interrupt structures, external buses, and so on. These are chosen based on the controller's final purpose. All the components within a microcontroller communicate via an internal system bus.

An embedded system usually contains one or more interconnected MCUs.

Most parts of an embedded system are electrical components. However, the mechanical structure or the housing of the sensor may also provide important functionality.

The range of applications for an embedded system are enormous. For example, an embedded system can be used as a closed-loop control system to monitor and process a system state, or adjust an output to maintain a desired set point such as temperature, speed, or direction. An embedded system can take over very specialized functions as part of a larger system, such as fault handling or networking, or it could be used to step through different stages of a program, based on environment and system requirements.

Let's look at a real-world example: a bike computer.

The bike computer senses wheel rotation and outputs the calculated values such as speed, distance, and time. Additional sensors, such as a heart rate monitoring sensor, could be connected to such a system.

The system is constrained primarily by size, cost, and battery life, which translates into power consumption. This means that the total system must be very cost- and power-efficient. Fortunately, size, weight, and cost are usually proportional, and the only counteracting variable may be the processor performance.

The bike computer, as a weak real-time system, should be able to process the input between two sensor impulses without any complex algorithms. This means that a very simple, low cost microcontroller can be used.

Another example of an embedded system is an engine control unit for a car.

In this example, both the functional requirements and the constraints are much greater than those of a bike computer. Spark and fuel injection timings need real-time calculations to ensure appropriate operation of the engine. Every computation must finish at a defined point in time. Otherwise, the engine will run sub-optimally and may even become damaged.

The system must also be reliable in harsh environments for relatively low cost, and have small device footprint due to limited installation space.

Many sensors and actuators have to be networked to the rest of the car and operated by the microcontroller. Consequently, a powerful, reliable, and high-performance microcontroller is needed.