Chapter 1

Introduction to Real-Time Embedded Systems

Introduction

- Real-time embedded systems have become pervasive.
- They are in your cars, cell phones, Personal Digital Assistants (PDAs), watches, televisions, and home electrical appliances.
- There are also larger and more complex real-time embedded systems, such as air-traffic control systems, industrial process control systems, networked multimedia systems, and real-time database applications.
- In fact, our daily life has become more and more dependent on real-time embedded applications.

Introduction (2)

- An embedded system is a microcomputer system embedded in a larger system and designed for one or two dedicated services. It is embedded as part of a complete device that often has hardware and mechanical parts.
- Examples, include the controllers built inside our home electrical appliances.
- Most embedded systems have real-time computing constraints. Therefore, they are also called real-time embedded systems.
- Compared with general-purpose computing systems that have multiple functionalities, embedded systems are often dedicated to specific tasks. For example, the embedded airbag control system is only responsible for detecting collision and inflating the airbag.

Introduction (3)

- Another difference is embedded systems may or may not have full-scale operating system support at all.
- Many small-sized embedded systems are designed to perform simple tasks and thus do not need operating system support.
- Embedded systems are reactive systems. They are basically designed to adjust a physical variable in response to the input signal provided by the end users or sensors, which are connected to the input ports.
- For example, adjusting the temperature of a furnace by adjusting the amount of fuel being injected into the furnace based on the difference between the desired temperature and the real temperature detected by temperature sensors.

Embedded systems classifications

- small-scale
- medium-scale
- large-scale.

Small-scale Embedded systems

- perform simple functions.
- usually built around low-end 8- or 16-bit microprocessors or microcontrollers.
- For developing the embedded software, the main programming tools are an editor, assembler, and integrated development environment (IDE).
- Examples of are mouse and TV remote control.
- They typically operate on battery. Normally, no operating system is found in such systems.

Medium-scale Embedded systems

- Have both hardware and software complexities.
- Use 16- or 32-bit microprocessors or microcontrollers.
- For developing embedded software, the main programming tools are C, C++, JAVA, Visual C++, debugger, source-code engineering tool, simulator, and IDE.
- They typically have operating system support.
- Examples are vending machines and washing machines.

Large-scale Embedded systems

- Have enormous hardware and software complexities.
- built around 32- or 64-bit microprocessors or microcontrollers, along with a range of other high-speed integrated circuits.
- Used for cutting-edge applications that need hardware and software code sign techniques.
- Examples are flight-landing gear systems, car braking systems, and military applications.

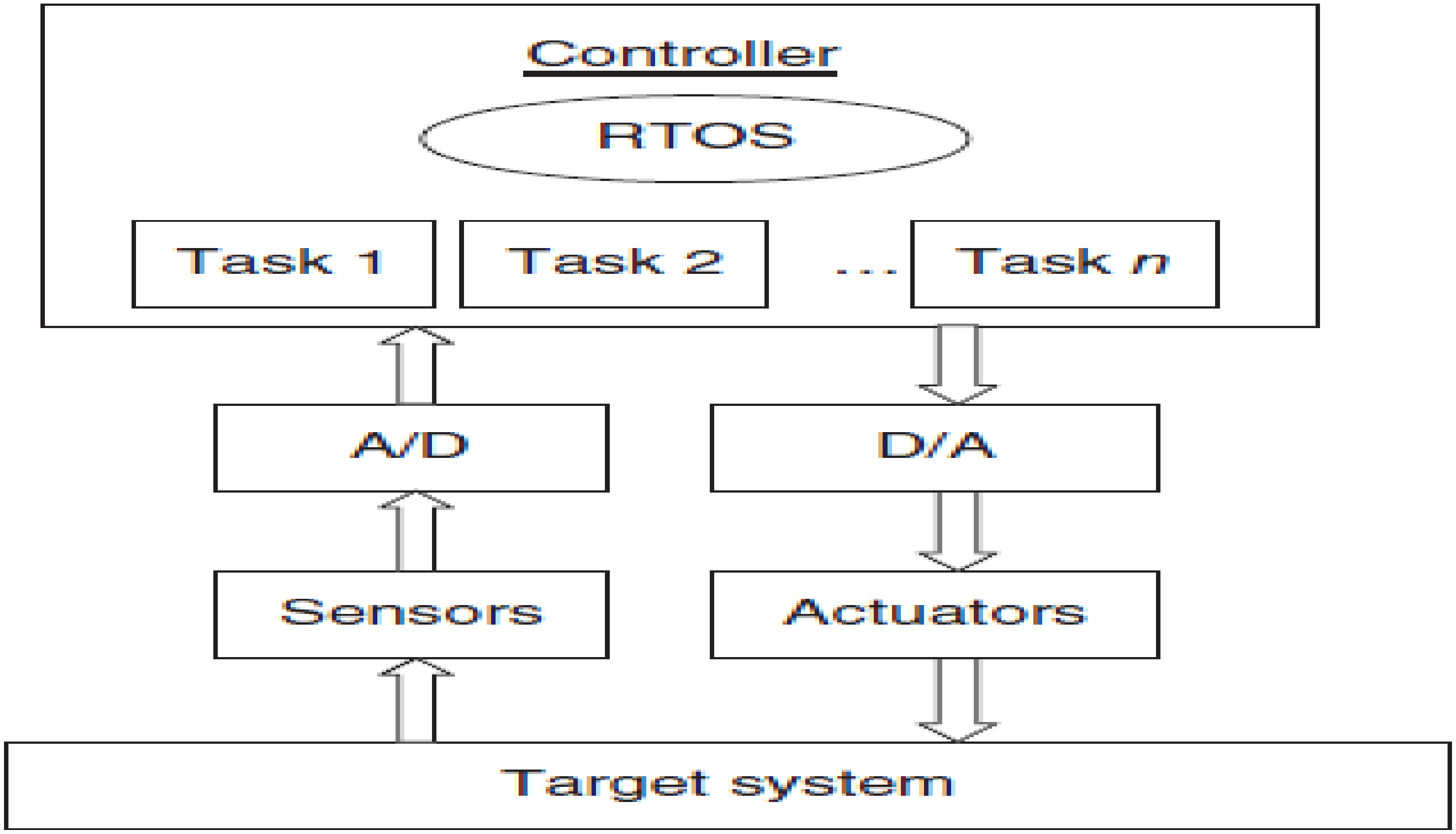
RT and Non-RT ES

- A non-real-time system is correctly designed and developed if it delivers the desired functions upon receiving external or internal triggers, with a satisfied degree of QoS. Examples are TV remote controls and calculators.
- Real-time systems are required to compute and deliver correct results within a specified period of time. A job of a real-time system has a deadline, being it hard or soft. Example, the airbag control system (DL = 0.1 Sec)

Real-Time Embedded System Characteristics

- System structure
- Real-Time Response
- Highly Constrained Environments
- Concurrency
- Predictability
- Safety and Reliability

System structure



Structure of real-time embedded systems.

System structure (1)

- It interacts with its environment continuously and timely.
- To retrieve data from its environment, the system must have sensors in place.
- In the real world, most of the data is characterized by analog signals. In order to manipulate the data using a microprocessor, the analog data needs to be converted to digital signals. Therefore, an analog-to-digit converter (ADC) is needed in between a sensor and a microprocessor.

System structure (3)

- The brain of an embedded system is a controller, which is an embedded computer composed of one or more microprocessors, memory, some peripherals, and a real-time software application.
- The software is usually composed of a set of real-time tasks that run concurrently, may or may not be with the support of a real-time operating system, depending on the complexity of the embedded system.

System structure (4)

- The controller acts upon the target system through actuators. An actuator can be hydraulic, electric, thermal, magnetic, or mechanic.
- The output that the microprocessor delivers is a digit signal, while the actuator is a physical device and can only act on analog input.
- Therefore, a digit-to-analog conversion (DAC) needs to be performed in order to apply the microprocessor output to the actuator.

Real-Time response

- A real-time system or application has to finish certain tasks within specified time boundaries.
- Moreover, the control law computing is also real-time: a cycle of a sensor data processing and control value computing must be finished before the next cycle starts; otherwise, the data to be processed will pile up.
- If a missile guidance system fails to make timely corrections to its attitude it can hit the wrong target.

Real-Time response (2)

- Deadlines of real-time tasks are typically derived from the required responsiveness of the sensors, actuators, and the dynamics of the target that the embedded system controls.
- “real-time” does not mean “real fast” or “the faster, the better.”
- Take a cardiac pacemaker as an example.

Highly Constrained Environments

- They are often run in highly resource-constrained environments.
- It make the system design and performance optimization challenging.
- Most embedded systems are constrained in terms of processor speed, memory capacity, and user interface.

Highly Constrained Environments (2)

- Many of them operate in an uncontrolled harsh environment. They have to survive excessive heat, moisture, vibration, shock, or even corrosion.
- Therefore, they must be optimized in terms of size, weight, reliability, performance, cost, and power consumption to fit into the computing environment and perform their tasks.
- Thus, embedded systems typically require far more optimization than standard desktop applications.

Concurrency

- Concurrency refers to a property of systems in which several computations are executing simultaneously and potentially interacting with each other.
- Embedded systems by design are closely interacting with their physical environment.

Concurrency (2)

- All the events have strict constraints on the response time. All deadlines should be met.
- Because of the existence of multiple control processes and each process may have its own control rate, many real-time embedded systems are multi-rate systems.
- For example, a real-time surveillance system needs to process both audio and video inputs, but they are processed at different rates.

Predictability

- A real-time system must behave in a way that can be predicted in terms of all timing requirements.
- For instance, it must be mathematically predictable if a specific task can be completed before a given deadline.
- Factors that go into this calculation are system workload, the power of processors, run-time operating system support, process and thread priorities, scheduling algorithm, communication infrastructure, and so on.

Predictability (2)

- They contain heterogeneous computing resources, memories, bus systems, and operating systems and involve distributed computing via global communication systems.
- Latency and jitter in events are inevitable in these systems. Therefore, related constraints should be specified and enforced
- Determinism represents the ability to ensure the execution of an application without concern that outside factors, such as unforeseen events, will upset the execution in unpredictable ways.

Safety and Reliability

- Safety means “freedom from accidents or losses” and is usually concerned with safety in the absence of faults as well as in the presence of single-point faults.
- Reliability, on the other hand, refers to the ability of a system or component to perform its required functions under stated conditions for a
- specified time.
- It is defined as a stochastic measure of the percentage of the time the system delivers services.

Hard and Soft Real-Time Embedded Systems

- In a hard real-time system most timing constraints are hard. In a soft one most timing constraints are soft.
- The system must meet A hard real-time constraint. If the deadline is missed, it will either cause the system failure or result in a zero usefulness of the delivered service.
- A soft constraint is a constraint that a system should meet, if the deadline is occasionally missed, it won’t cause any disastrous result, and the delivered service is still useful to a certain extent.