Chapter 2
Features of Embedded System

Abstract This chapter will introduce the basic elements of embedded systems (or dedicated systems). The integrated control systems represent one of the areas of modern electronics which most important and developed with a variety of application in many fields from biomedical to the automotive.

2.1 The Components of Embedded System

An embedded system (Fig. 2.1) is, basically, a computer controlled device designed to perform specific tasks. In most cases, these tasks help revolve the real-time control of machines or processes. Embedded systems are cheaper than general purpose system, such as PCs [1, 2].

2.1.1 Processor

The main part of an embedded system is the processor, which could also be a generic microprocessor or a microcontroller and programmed to perform the specific tasks for which the integrated system has been designed.

2.1.2 Memory

Electronic memory is an important part of embedded systems and three essential types of memory can be described: RAM, or random access memory, ROM, or read only memory, and Cache. The RAM is one of the hardware components where data are temporarily stored during execution of the system. The ROM contains input-output routines that are needed for the system at boot time. The cache, instead, is used by the processor as a temporary storage during the processing and transferring of data.

M. Di Paolo Emilio, *Embedded Systems Design for High-Speed Data Acquisition and Control*, DOI 10.1007/978-3-319-06865-7_2

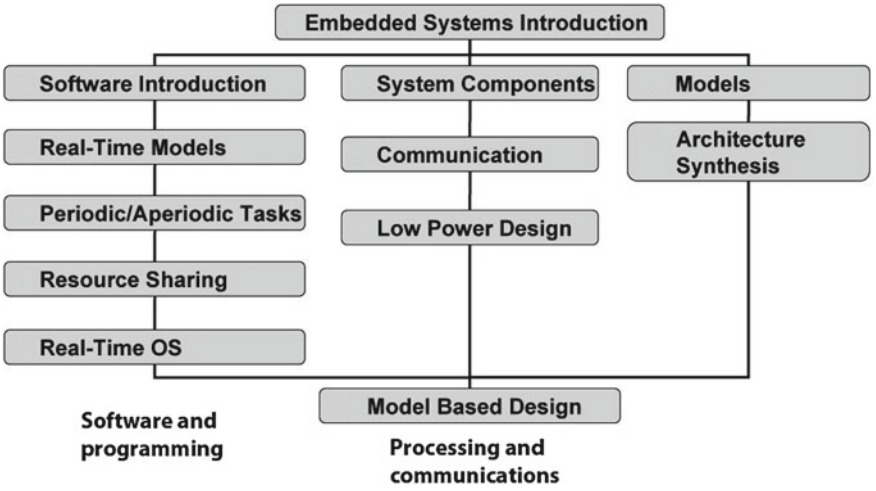

Fig. 2.1 Embedded system design

2.1.3 System Clock

The system clock is used for all processes is running on an embedded system and requires precise timing information. This clock is generally composed of an oscillator and some associated digital circuitry.

2.1.4 Peripherals

The peripheral devices are provided on the embedded system boards for an easy integration. Typical devices include serial port, parallel port, network port, keyboard and mouse ports, a memory unit port and monitor port. Some specialized embedded systems also have other ports such as CAN-bus.

2.2 Characteristics and Example of Embedded System

Most embedded systems are designed to perform an continued action at a low cost. Most of these systems also have constraints on the performance in terms of hardware and software, such as require operating in real time when a system needs high sped while executing some functions, but may tolerate lower speed for other activities. It is difficult to characterize the speed or the cost of an generic embedded system, especially for systems that have to process a large quantity of data. Fortunately,

most of the embedded systems have the essential characteristics that can be designed with a combination of hardware and high-performance software. To get an idea, just think of a decoder for a satellite television. Although a system should process tens of megabits of data per second, most of work performed by dedicated hardware, separates rule and decoding of the data flow in multi-channels digital video output. Embedded CPU calculated the locations of the data in the system, manages interrupts and clock systems [3, 4].

Usually, the hardware of an embedded system must comply with the performance requirements much less stringent in according to the hardware of the primary system itself. This allows that the architecture of an embedded system, for example, must be intentionally simplified and compared to that of a general-purpose computer with the same tasks, using a CPU more economic that basically behaves well for these secondary functions.

In the case of portable systems, costs reduction becomes a priority. This kind of system, in fact, often are made by a highly integrated CPU, a chip dedicated to all other functions and a single board of memory. In this case each component is selected and designed to reduce as much as possible the costs. The useful software to manage many embedded systems is called firmware. The firmware is a type of software that, for example, can be found in ROM or Flash memory chips. The software and firmware are designed and tested with much more attention than traditional software for personal computers.

Many embedded systems avoid incorporating components with moving parts (such as hard disk) that are less reliable than solid-state components such as flash memory.

Moreover, embedded systems may not be physically accessible (e.g. space systems); therefore, the system must be capable of a self-reset in case of data loss or corruption. This feature is very often obtained with the addition of a component called Watchdog that resets the computer in regular time intervals by an internal timer.

In the design of a modern and reliable embedded system is possible to note two fundamental characteristics: reprogrammability and the dimension. In fact, it would be helpful to think of a dedicated system can be readapted if, for example, system upgrade is required. Embedded systems are a classical discrete elements of ASIC design that allow the advanced optimization because the hardware occupies the necessary space strictly, making the control system easily integrated. An application-specific integrated circuit (ASIC) is an integrated circuit (IC) that has been customized for purpose use. Today the control of vehicles is one of the main applications of embedded systems. In a single high-end car you can find hundreds of embedded systems called ECU (Electronic Control Unit), physically distributed in the vehicle and connected to the different internal networks (networks intra-vehicle) specially designed, in most cases with stringent requirements of ‘quality of service’. A computer is the first and foremost versatile: it can be programmed to suit various areas of application. Conversely, the embedded system is a device dedicated to the performance of a single task, or a very narrow class of tasks. Thanks to the specificity of the run application, the embedded system can be designed to optimize particular

of cost and performance. The general purpose of the computers is designed with standards and reference architectures; and vice versa is difficult to define standards for embedded systems because each specific application leads to different design choices. Typical functions of embedded systems can be the following:

- Processing: ability to process the analog/digital signals.
- Communication: ability to transfer signals ("Information") from/to the outside world.
- Storage: the ability to preserve the temporary information within the embedded system.
- Each specific application made by an embedded system has different requirements for processing, power suppy, storage and communication.
- A same functionality (e.g., the ability to acquire still images via a CCD sensor) can be optimized radically in different way when applied, for example, a digital camera or a cell phone or a digital camcorder.

Morever, commercial features of embedded systems can be described in the following points:

- Final Cost: The cost of the final product is a very important parameter for the design choices.
- Time to market: in the design of an embedded system must always keep in mind the timing you want the product listed on the market; taking too long to design a device means that it's difficult to overcome the fast changes in the market.
- Life time: Another important factor is the expected lifetime for the product; which can varied from a few days to several years or decades.
- Volume: the quantity of stock planned for the system is one important factor in the design phase.

Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. Hardware and software characteristics of embedded systems can be described in the following points:

- Communication interfaces: typically the sale price of an embedded system is low, the choice of communication interfaces is critical because it greatly affects the final price of the product.
- User Interface: In many embedded systems the user interface consists of a few buttons and/or LEDs; in others, it uses the user interface of a host system.
- Power management: is a crucial factor to be considered for all embedded systems are powered by batteries.
- Dimensions and weight: in many cases, the physical characteristics are another critical factor; usually the embedded system must be small, very light or with a particular form (for example, very thin).
- Quality of service: many applications of embedded systems have stringent requirements in terms of QoS (Quality of Service); as a particular case, many applications require the provision of services in real time with stringent timing constraints.

- Code size: the storage capacity of embedded systems is limited, so the size of the internal program (e.g. firmware) is a important factor.
- Numeracy/Communication skills/Storage: commensurate to the specific application performed by the embedded system.
- Updating the program: it is useful to include the ability to update the programs in embedded systems so as to correct errors discovered after the production and introduce new features.

In addition to the parameters involved in the market, the hardware, software features and embedded systems are used to be dependable. Actually, in the design phase is necessary to consider the following aspects:

- Reliability: realistic assessment of the probability that the system fails.
- Maintainability: the system can be repaired or replaced within a certain time interval.
- Availability: probability that the system is working; essentially depends on the reliability and maintainability.
- Safety: properties related to the possibility that in the event of system failure are caused damages to people or things.
- Security: resilience of the system against unauthorized use.

To design the embedded systems, it should take into account aspects such as the speed of development, the economy of scale, maintainability and so on. Consequently, it is not possible to develop the hardware also without consider the software design. If the embedded system is safety-critical, the choice of the software organization plays a crucial role in the ability to certify the system for the use to which it is intended.

The real-time system is a system designed to operate within the well-defined time parameters. Practically, a real-time system operates correctly only if every input configuration is produced by the right output respecting well-defined time constraints.

2.3 Hardware and Software Design

The device required to achieve by designing an embedded system is, certainly, a system that will optimize various metrics of design; the most common are as the following [1–4]:

- Unit cost and cost NRE (non-recurring costs).
- Size and weight.
- Performance and power consumption.
- Flexibility and maintainability.
- Time-to-market.
- Correctness.
- Security of the system.

Usually, these metrics are in contrast between each other and become necessary to choose a compromise, esteeming and considering in an accurate way. Crucial importance is the metrics of cost/performance/power consumption and those of time-to-market/NRE. The relationship between time-to-market and NRE is very crucial to avoid entering in the market too late; If you have knowledge that the units produced will be few, it will be preferable to have lower NRE costs because it will not be possible to cover the next phase of the sale. It is essential to design an embedded system based on the constraints imposed by the metric, for this we use a methodology that uses estimators and cost modeling.

The constraints required for certain applications are more and more stringent, response times, size, weight and low power consumption. The flow of prototyping is often very complex and becomes important the reuse of pre-designed and tested hardware/software components is to reduce the time-to-market and NRE costs.

There are many independent techniques for estimating the costs: analogy, top-down, BottomUp, parametric; is preferable to use more than one and combine the results. Software parameter models (COCOMO) and Hardware (FPGA) allow to estimated the metrics automatically. The applicable cost to the modeling and design the process is called RASSP (Rapid Prototyping of Application Specific Signal Processors) and can use different methodologies.

The platform on which an embedded system can be developed varies drastically and depends on its complexity, utilities (electrical), cost, and scope of use: going from the PLC and microcontrollers to more complex architecture based on sophisticated integrated circuits (System-on-a-chip, SoC). The SoC enclose, in a single ASIC integrated circuit, microcontroller/CPU and/or DSP, memory, and clock oscillator, voltage regulator, any interfaces A/D, D/A, and external port (USB, ethernet, etc.). Some development platforms (reference board and reference design) are widely used based on ARM, MIPS, Coldfire/68k, H8, SH, V850, FR-V, M32R, and so on, for example IBM-compatible architectures with X86 or PowerPC CPU. Other architectures are based on PICmicro, Intel 8051 and Atmel AVR microcontrollers. Another common design method involves the use of FPGAs (Field-Programmable Gate Array), with the programming of all the internal logic, including the CPU. Typically for interfacing you will use the FPGA with other integrated circuits. This situation is in contrast with desktop computer market, which currently consists of only a few competing architectures, mainly the Intel/AMD X86 and PowerPC Apple/Motorola/IBM.

It's also useful to disclose the standard PC/104, dealing only with the form factor (the size of the motherboard) and the communication bus. It is typically employed in the industrial desktop systems (X86 CPU) with adaptations for specific uses. The user interfaces for embedded systems are also various between systems and therefore deserve some additional comment.

The designers of interfaces such as Apple Computer and HP tend to minimize the number of different user interactions. For example, their systems using only two buttons (the absolute minimum) to control a menu system. A touch screen or buttons to the edges of the screen can be also used to minimize the interaction with the user. As with other software, embedded system designers use compilers, assembler and debugger for developing the software supplied in the system.

An in-circuit emulator (ICE) is a hardware device that replaces or interfaces with the microprocessor, and provides functionality to load and debug test within the system. To speed up, make diagnostics and debugging, especially on large scale systems manufactured, is integrated at the level of SoC, microcontroller, or CPU, a JTAG interface (IEEE 1149.1): a standard simple interface and inexpensive that suspends the normal mode of the process and interrogate the phases by connection to a personal computer.
Some utilities add a redundancy check (checksum) or a Cyclic Redundancy Check (CRC) to the program, in order to allow the embedded system to check the validity of the program.
For systems using Digital Signal Processor (DSP), designers can use a tool such as MathCad or Mathematica to simulate the mathematics.
Compilers and specific linker can be used to improve the optimization of hardware. An embedded system may have its own specific language or development program or offer improvements to an existing language.
The programs for the generation of the software may have different origins: companies specialized in embedded system market, GNU project, development tool for personal computer if the embedded processor is very similar to a common PC processor.
The presence or absence of a complete operating system on an embedded system depends on its architectural complexity and the range of use. In the simplest case the embedded devices might be without an operating system itself. On simple microcontrollers typically operate cyclically few bytes of a single program, sometimes referred as a program of "monitor", dedicated mainly to monitor the status of the ports I/O. Complex environments can be applied to the same operating systems commonly used for the general purposes (Linux, Windows CE etc.), possibly custom (to operate in an environment with minimal resources), or more specialized to handle events of real-time operating systems (such as VxWorks and QNX), or highly specialized and not available on the market.

References

1. Heath, S. (1997). *Embedded systems design*. Oxford: Newnes.
2. Vahid, F., & Givargis, T. (2002). *Embedded system design: A unified hardware/software introduction*. Hoboken: Wiley.
3. Wilmshurst, T. (2008). *Designing embedded systems with PIC microcontrollers*. Oxford: Newnes.
4. (2008). *The art of designing embedded systems*. Oxford: Newnes.

http://www.springer.com/978-3-319-06864-0

Embedded Systems Design for High-Speed Data Acquisition and Control

Di Paolo Emilio, M.

2015, XXII, 155 p. 123 illus., 5 illus. in color., Hardcover
ISBN: 978-3-319-06864-0