Chapter 2

BASIC CONCEPTS OF REAL TIME OPERATING SYSTEMS

Franz Rammig, Michael Ditze, Peter Janacik, Tales Heimfarth, Timo Kerstan, Simon Oberthuer and Katharina Stahl

Abstract

Real-time applications usually are executed on top of a Real-time Operating System (RTOS). Specific scheduling algorithms can be designed. When possible, static cyclic schedules are calculated off-line. If more flexibility is needed on-line techniques are applied. These algorithms are bound to priorities which can be assigned statically or dynamically. Designing a proper RTOS architecture needs some delicate decisions. The basic services like process management, inter-process communication, interrupt handling, or process synchronization have to be provided in an efficient manner making use of a very restricted resource budget. Various techniques like library-based approaches, monolithic kernels, microkernels, or virtual machines/exokernels are applied, based on specific demands. Safety critical application can be supported by separation of applications either in the time or the space domain. Multi-core architectures need special techniques for process management, memory management, and synchronization. The upcoming Wireless Sensor Networks (WSN) generate special demands for RTOS support leading to dedicated solutions. Another special area is given by multimedia applications. Very high data rates have to be supported under (soft) RT constraints. Based on the used encoding techniques (e.g. MPEG) dedicated solutions can be created.

Keywords: RTOS, Scheduling, Safety Critical Systems, Wireless Sensor Networks
2.1 Introduction

Most embedded systems are bound to real-time constraints. In production control the various machines have to receive their orders at the right time to ensure smooth operation of a plant and to fulfill customer orders in time. Railway switching systems obviously have to act in a timely manner. In flight control systems the situation is even more restrictive. Inside technical artifacts many operations depend on timing, e.g. the control of turbines or combustion engines. This is just a small fraction of such applications. Even augmented reality systems are real-time applications as augmenting a moving reality with outdated information is useless or even dangerous.

“Real-time” means that the IT system is no longer controlling its own time domain. Now it is the progress of time of the environment which dictates how time has to progress inside the system. This environmental time may be the real one of our physical world or it may be artificially generated by some surrounding environment as well. For the embedded system there is no difference between these options. Kopetz defines real-time systems as “A real-time computer system is a computer system in which the correctness of the system behaviour depends not only on the logical results of the computation, but also on the physical instant at which these results are produced” [Kop97]. This means that in strict real-time systems a late result is not just late but wrong. The meaning of “late” of course has to be defined dependent on the specific application. In case of an air-bag controller it is intuitively clear what real-time means and it is easy to understand that a late firing of the air-bag is not only late but definitely wrong.

It can be concluded that in real-time systems the program logic of application tasks has to be augmented by information about timing. Such timing information contains the earliest point of time the task may be started as well as the latest allowed finishing time. This, together with the program logic may be seen as a specification for the computing system what to do and when to do it.

Many such tasks may have to be executed concurrently on an embedded computing system. Such situations usually are handled by some kind of operating system. The same is true in case of real-time systems. But now an additional objective function is introduced, an objective function which dominates most other ones: Formulated real-time constraints have to be respected. An operating system which is capable of taking care of this is called a “Real-time Operating System (RTOS)”. Of course some additional information is needed by an RTOS to manage real-time tasks. Especially the worst-case execution time (WCET) on the specific target architecture of any real-time task has to be available. Determining the WCET of a task is a demanding goal on its own. It must never be underestimated. On the other hand the potential