REAL TIME SYSTEM DESIGN

SHEM-TOV LEVI
ASHOK K. AGRAWALA

REAL-TIME SYSTEM DESIGN

Shem-Tov Levi

Ashok K. Agrawala

McGraw-Hill Publishing Company

New York St. Louis San Francisco Auckland Bogotá Caracas Hamburg
Lisbon London Madrid Mexico Milan Montreal New Delhi Oklahoma City Paris
San Juan São Paulo Singapore Sydney Tokyo Toronto

This book was set in Times Roman by Publication Services, Inc.
The editor was David M. Shapiro;
the production supervisor was Friederich W. Schulte.
The cover was designed by John Hite.
Project supervision was done by Publication Services, Inc.
R. R. Donnelley & Sons Company was printer and binder.

REAL-TIME SYSTEM DESIGN

1 2 3 4 5 6 7 8 9 0 DOC DOC 9 5 4 3 2 1 0

ISBN 0-07-037491-0

Library of Congress Catalog Card Number: 90-60025.

LIST OF FIGURES

PREFACE

Computers have been used for real-time applications for a long time. The design and implementation of such systems has usually been carried out as an extension of the system design principles used for general purpose computer systems. In particular, real-time systems have often been designed as interrupt-driven systems with priority-based scheduling. Priority structures are used to accomplish the real-time processing by assigning higher priority to critical tasks. In this approach *time* is not treated explicitly in resource management or in scheduling of tasks.

The hard real-time applications of tomorrow must provide support for reliable distributed operation. The past design methodologies may not be adequate to meet these challenges. In this book we have attempted to present a comprehensive approach to the design of the next generation of real-time systems.

We believe that it is essential for the next generation of hard real-time systems to use time directly and explicitly. The approach taken here is to make a uniform representation and use of time. Using an object-oriented approach, we consider time to be an integral property of every entity in the system. In addition, a hard real-time system has to have a deterministic and predictable behavior. The techniques useful for making the system behavior deterministic are also presented.

It has often been noted that the maintenance of a complex real-time system becomes a major problem. Any change usually requires extensive testing to assure functional as well as temporal correctness. The approach we propose is to carry out a verification of the resource allocation. In this way we can reduce significantly the testing requirements. The ideas presented in this book are realistic, they have been implemented, and they are therefore practical.

We present a detailed discussion of the current state of the art of real-time systems and application methodologies along with some major recent advances.

This includes a complete presentation of the design, implementation, verification, and testing issues of the real-time problems. Both the application level and the system level view are taken.

This book is aimed at the professional in the field who has to deal with real-time systems from different perspectives. It is useful to the researchers as it presents many novel ideas, not only for real-time systems but also for distributed, reliable systems. The complete treatment of real-time systems makes it well suited for a graduate or advanced undergraduate course on the subject.

This book reflects our experience in building and implementing real-time systems that are in use extensively, as well as our backgrounds in academic research. The main motivation for this work was to explore ways of improving the current design methodologies to meet the challenges of tomorrow. As we found no adequate text on the subject, we started compiling our notes, which resulted in this book.

While writing the book we found many issues we wanted to share with our readers. Some of these issues concern general subjects such as fault tolerance or complexity, while others concern practical issues such as debugging. A detailed presentation of all these topics would make the book unwieldy. The more information we add the more difficult it would be to focus. We therefore decided on this version of the book, in order to present a discussion of all relevant topics, with suitable references for further study.

As the book is aimed at various types of professionals in the field of real-time systems, let us suggest a structure of the book according to reading interests.

The book has sections that are common to all readers, as they include basic information as well as notation and semantic definitions. These sections are: Section 2.1, Section 2.2, Section 2.3, Section 3.1, Section 4.1, Section 5.2, Section 6.1, Section 6.2, Chapter 11, Chapter 12, Section 13.1, Section 14.1, Section 15.1, Section 15.4, Section 16.1, and Section 17.1. The information introduced in these sections is essential for understanding the rest of the book.

Readers who are more engineering-oriented may want to read the following sections: Section 2.4, Section 2.5, Section 3.2, Section 3.3, Section 4.2, Section 5.1, Section 7.1, Chapter 9, Chapter 10, Section 13.2, Section 13.3.1, Section 13.4.1, Section 14.2, Section 14.4, Section 15.2, Section 15.3, Section 16.2, Section 17.2, and Section 17.3.

Readers who are more research-oriented may find further interest in the following sections: Section 4.3, Section 6.3, Chapter 7, Chapter 8, Section 13.2, Section 13.3, Section 13.4, Section 14.2, Section 14.3, Section 14.4, and Section 17.3.

We hope that the above suggested structure will help each professional to focus easily where he or she finds interest.

ACKNOWLEDGMENTS

A project of this magnitude is clearly the result of a lot of support from many friends. Members of the Systems Design and Analysis Group of the Department

of Computer Science, University of Maryland have actively contributed to the development of the material in the book and are implementing the **MARUTI** operating system, which is based on the ideas presented in this book. We would like to gratefully acknowledge the support for our research that has contributed to the development of this book. Our research was supported by the Office of Naval Research and Rome Air Development Center through the Army Strategic Defense Command through grants and contracts to the Department of Computer Science at the University of Maryland. We would also like to acknowledge the Israel Aircraft Industries, Ltd., for contributing to the making of this work.

Shemi-Tov Levi
Ashok K. Agrawala

CONTENTS

Part III Real-Time Operating Systems

CHAPTER 1

INTRODUCTION

The ever-increasing use of computer systems is clear evidence that the functional capabilities provided by them can be used very effectively for a variety of purposes and in a large number of fields. In many of these applications, the performance of the computer system is measured with metrics such as response time or turnaround time, the implication being that the faster the better with no specific requirement being placed on the timing behavior of the system. *Real-time applications* are different from this paradigm of computation in that they impose strict requirements on the timing behavior of the system. The systems that support the execution of real-time applications and ensure that the timing requirements are met are often referred to as *real-time systems*. Traditionally, the correctness of many computer systems has been taken to imply their logical and functional behavior. For real-time systems correctness depends on the temporal properties of this behavior as well.

As the price and performance of digital computers continue to improve and their size, weight, and power requirements continue to decrease, there has been a steady increase in the use of computer-based real-time systems in a wide variety of fields. Application domains such as military, industry, and medicine indicate a wide spectrum of possible implementations. Current real-time system examples include nuclear power plant control, industrial manufacturing control, medical monitoring, digital fly-by-wire avionics, weapon delivery systems, space navigation and guidance, and reconnaissance systems. As the use of real-time systems has spread, the timing requirements have become more stringent and the reliability requirements more difficult to achieve.

In general, we call a system a real-time system when it can support the execution of applications with time constraints on that execution. A variety of systems clearly meet this definition. Note that no assumptions are made about the structure or the architecture of the computer system used. A particular class of such systems comprises embedded computer systems.

1.1 EMBEDDED COMPUTER SYSTEMS

Many complex systems in use today require a very elaborate control and computational facility to support their continued proper functioning. Such systems often use dedicated hardware as controllers. Clearly, all the computations and control functions can also be carried out by an appropriate computer system. When a computer system is used in a large system to provide control and computation functions, it is often referred to as an *embedded computer system*. Currently we find such systems in almost every aspect of our lives, with computers being introduced into new systems at an ever-increasing rate.

An embedded computer system has to manage and control the rest of the system. It collects data through sensors and issues control commands to mechanical, electromechanical, and electronic actuators. Figure 1.1 illustrates a typical system of this class. Note that this figure could also depict any process control system; we are using it to convey the idea of a computer as a controller in such a system.

A distinguishing feature of embedded computer systems is that they usually provide an execute-only environment, in which no program development goes on. The processing requirements in these systems do not change as they handle a fixed and well-defined workload. Although some embedded systems are designed

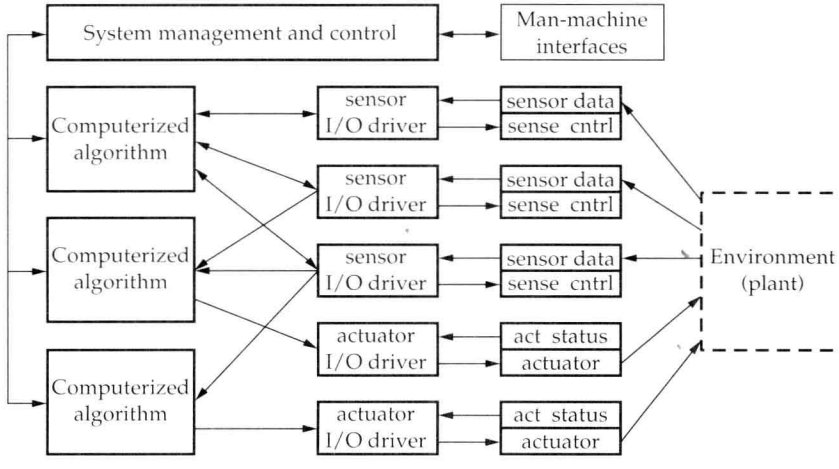

FIGURE 1.1
Embedded computer system.

to handle transient inputs, the processing requirements for such transient requests are predefined. For example, the embedded computer system used in automobiles to control the fuel injection and spark ignition executes one fixed program that may adjust its parameters as its sensors provide it with new information about environmental conditions.

When we study embedded computer systems, their close interaction with mechanical, electromechanical, and electronic components requires that such components be considered as a part of the system and that the expressions and modeling tools used in the study have the capability of representing the interdisciplinary properties of the system. We cannot describe the performance of a robot arm without reference to its mechanics, electronics, and control software. Furthermore, each discipline must maintain consistency with the others. We want the system description to resolve interdisciplinary contradictions when they exist. For example, consider a control system with high-bandwidth control software and control hardware that is very slow. The system description of this example must allow for combining the hardware and its control software. The description must reflect the overall slow response time of this system.

Specifying a system that makes use of techniques from several disciplines starts at the level of the whole system. Out of this high-level specification, we specify the subsystems up to single-discipline components. Derivation of subsystem specifications based on the high-level specification must be consistent and unambiguous. Budgeting software subsystem specifications must, therefore, not only support the performance description of the whole system; it must also be derivable from a whole system description and allow verification of the system properties.

1.2 HISTORICAL PERSPECTIVE

Real-time systems developed from embedded computer systems are still an important family of real-time systems. The use of digital computers in such systems started with the replacement of analog processing sections of control systems during the 1950s. A major step in this direction was taken in March 1956. TRW engineers were contracted by Texaco to computerize a process control system in a polymerization unit of a refinery in Port Arthur, Texas [11]. The process controller employed an RW-300 computer, which controlled 26 flows, 72 temperatures, and 3 pressures. It was announced to be operational in March 1959. In 1962, another major step was taken in the chemical industry, at ICI in England, where a single Ferranti computer replaced complete analog instrumentation that controlled 129 valves and measured 224 nodes.

Other examples of the early use of embedded systems exist. Airborne controllers were needed for missile and aircraft applications. NASA engineers developed many flight control systems during the 1960s for the Mercury, Gemini, and Apollo projects. Other military projects employed time-driven sequencers, digital comparators, and adders for flight control subsystems.